

Graph-based Agent Memory: Taxonomy, Techniques, and Applications

Chang Yang[†], Chuang Zhou[†], Yilin Xiao[†], Su Dong, Luyao Zhuang, Yujing Zhang, Zhu Wang
 Zijin Hong, Zheng Yuan, Zhishang Xiang, Shengyuan Chen[‡], Huachi Zhou[‡], Qinggang Zhang[‡]
 Ninghao Liu, Jinsong Su, Xinrun Wang, Yi Chang, Xiao Huang

Abstract—Memory emerges as the core module in the Large Language Model (LLM)-based agents for long-horizon complex tasks (e.g., multi-turn dialogue, game playing, scientific discovery), where memory can enable knowledge accumulation, iterative reasoning and self-evolution. Among diverse paradigms, graph stands out as a powerful structure for agent memory due to the intrinsic capabilities to model relational dependencies, organize hierarchical information, and support efficient retrieval. This survey presents a comprehensive review of agent memory from the graph-based perspective. First, we introduce a taxonomy of agent memory, including short-term vs. long-term memory, knowledge vs. experience memory, non-structural vs. structural memory, with an implementation view of graph-based memory. Second, according to the life cycle of agent memory, we systematically analyze the key techniques in graph-based agent memory, covering memory extraction for transforming the data into the contents, storage for organizing the data efficiently, retrieval for retrieving the relevant contents from memory to support reasoning, and evolution for updating the contents in the memory. Third, we summarize the open-sourced libraries and benchmarks that support the development and evaluation of self-evolving agent memory. We also explore diverse application scenarios. Finally, we identify critical challenges and future research directions. This survey aims to offer actionable insights to advance the development of more efficient and reliable graph-based agent memory systems. All the related resources, including research papers, open-source data, and projects, are collected for the community in <https://github.com/DEEP-PolyU/Awesome-GraphMemory>.

Index Terms—Agent, Multi-Agent System, Agent Memory, Knowledge Graph, Self-Evolving, Graph-based Memory

I. INTRODUCTION

The past few years witnessed the rapid development of Large Language Model (LLM)-based agents, which have demonstrated remarkable success in complex, long-horizon tasks across diverse domains, from software engineering [1] and

[†]Equal contribution.

[‡]Corresponding authors: Qinggang Zhang, Huachi Zhou, Shengyuan Chen.

Chang Yang, Chuang Zhou, Yilin Xiao, Su Dong, Luyao Zhuang, Yujing Zhang, Zhu Wang, Zijin Hong, Zheng Yuan, Shengyuan Chen, Huachi Zhou, Qinggang Zhang, Ninghao Liu, and Xiao Huang are with the The Hong Kong Polytechnic University, Hong Kong SAR, China (e-mail: {chang.yang, chuang-qzj.zhou, yilin.xiao, su.dong, luyao.zhuang, yu-jing.zhang, juliazhu.wang, zijin.hong, yzheng.yuan, huachi.zhou}@connect.polyu.hk, {sheng-yuan.chen, qinggang.zhang, ninghao-prof.liu, xiao.huang}@polyu.edu.hk).

Jinsong Su and Zhishang Xiang are with the School of Information, Xiamen University, China (e-mail: xiangzhishang@stu.xmu.edu.cn, jssu@xmu.edu.cn).

Xinrun Wang is with the School of Computing and Information Systems, Singapore Management University, Singapore (e-mail: xrwang@smu.edu.sg).

Yi Chang is with the School of Artificial Intelligence, Jilin University, Changchun, China (e-mail: yichang@jlu.edu.cn).

mathematical reasoning [2] to multi-agent tasks [3] and open-world exploration [4]. The inherent language understanding, generation, and inference capabilities of LLMs enable LLM-based agents to autonomously perceive environments and make decisions, which reduces manual intervention and reshapes the paradigm of intelligent systems [5].

Despite notable advancements, LLM-based agents are still constrained by the intrinsic limitations of LLMs. (i) Knowledge cutoff: LLMs are trained on static datasets with fixed time boundaries, resulting in knowledge cutoff issues that prevent them from incorporating real-time information (e.g., current financial data) or domain-specific knowledge beyond their pre-training corpora. This limitation undermines their ability to adapt to dynamic environments and open-ended scenarios. (ii) Tool incompetence: Although tool use represents a core capability of LLM-based agents [6], [7], existing LLMs demonstrate limited capacity for efficiently learning and applying novel tools, which substantially constrains the agent performance. (iii) Performance saturation: LLM-based agents exhibit persistent failures in iterative, long-horizon tasks due to their inability to accumulate task-specific insights and leverage historical experiences for refining decision strategies across extended interactions. Consequently, agents may repeatedly commit similar errors without demonstrating learning behavior to correct the errors for successful task completion.

To address these challenges, memory [8] has emerged as a critical component for advancing LLM agents towards four key objectives: i) **Personalization and Specification**. [9]: Memory enables agents to capture user preferences, interaction histories, and task-specific contexts for tailored responses, such as remembering workflow habits in software engineering or communication styles in conversational scenarios. Memory bridges general knowledge with specific context, storing both universal facts and particular histories to ground responses in personalized, context-aware information [10]. ii) **Long-term Reasoning beyond Context Window**. While LLMs operate within finite context windows with static parametric knowledge, memory systems provide unbounded external storage that enables continuous learning and adaptation. Memory allows agents to retain information across extended temporal horizons, accumulate post-deployment experiences including successes and failures, and dynamically refine strategies without model re-training. iii) **Self-improving** [11]: By accumulating experiential knowledge, reasoning patterns, and feedback, agent memory supports iterative enhancement of adaptability and performance, thus enabling the self-improving of LLM-based Agents on

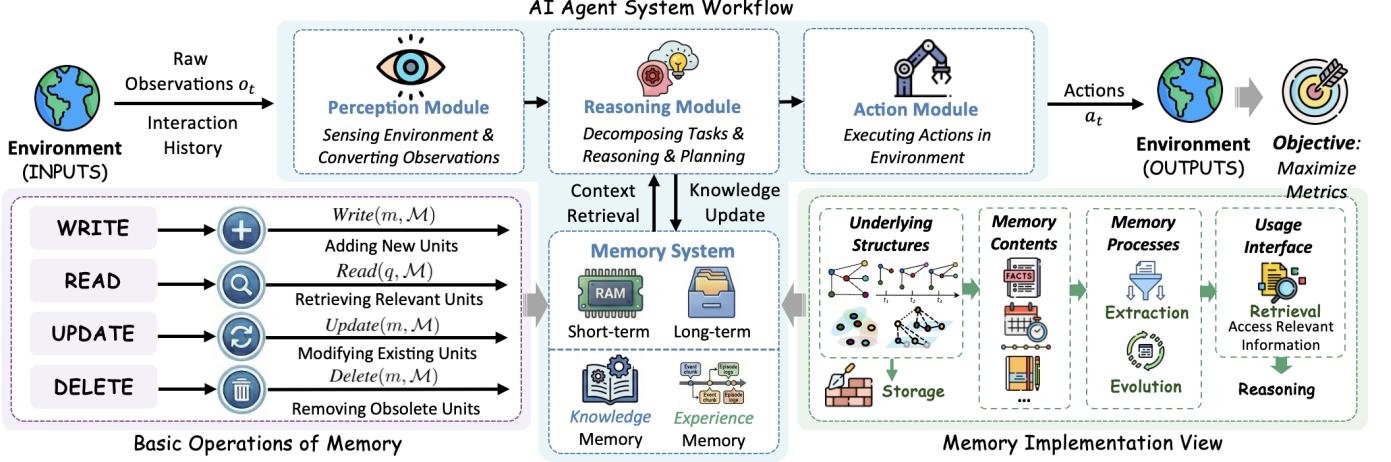


Fig. 1: A diagram illustrating the workflow of an AI agent system, and the detailed implementations of the memory system.

the tasks without parameter updating. iv) **Hallucination mitigation** [12]: Grounding outputs in structured, verifiable memory content reduces reliance on potentially unreliable parametric knowledge. In essence, memory transforms stateless reactive models into stateful adaptive entities capable of relationship building, trajectory-based learning, and increasingly sophisticated personalized behavior.

Traditional implementations of agent memory primarily adopt linear, unstructured, or simple key-value storage paradigms, such as fixed-length token sequences, vector databases, and log-based buffers [13], [14]. While these frameworks enable basic information storage and retrieval, agent memory demands more complex functionalities, such as relational modeling, hierarchical organization, and causal dependencies. Graph-based agent memory [15], [16] has emerged as the frontier for 2025–2026 research, transitioning from a passive ‘log’ of facts to a structured topological model of experience that preserves how information is connected over time. Unlike traditional linear or unstructured memory, graph-based memory can naturally encode relational dependencies between memory elements due to its intrinsic ability to model entity relationships, capture hierarchical semantics, and support flexible traversal and reasoning. Even plain memory can be regarded as a degenerate graph with trivial relationships, positioning graph-based agent memory as a general and flexible framework for agent memory design. Recently, there has been a surge of research into graph-based memory architectures for LLM agents, including knowledge graphs (KG), temporal graphs, hypergraphs, hierarchical trees/graphs and hybrid graphs [17], [18], which demonstrated the efficacy across diverse scenarios, such as hierarchical task planning, multi-session conversational understanding, and neuro-symbolic reasoning.

Therefore, we present a comprehensive survey that consolidates the state-of-the-art in graph-based agent memory, categorizes their core techniques, synthesizes their applications, and identifies open challenges. Our contributions are fourfold:

- We propose a taxonomy of agent memory, including short-term vs. long-term memory, knowledge vs. experience memory, non-structural vs. structural memory, with an

implementation view of graph-based memory (Section III).

- We systematically analyze the critical memory management techniques, covering memory extraction (Section IV), memory storage (Section V), memory retrieval (Section VI) and memory evolution (Section VII).
- We summarize open-sourced libraries and benchmarks (Section VIII) that support the development and evaluation of self-evolving graph-based agent memory across diverse application scenarios (Section IX).
- We identify critical challenges and outline future research directions to advance efficient and reliable graph-based agent memory systems (Section X).

This survey aims to provide a holistic overview of graph-based agent memory, offering valuable insights for researchers to advance memory design and enabling practitioners to select appropriate structures and techniques for specific applications.

II. PRELIMINARIES

Definition II.1 (AI Agents). *An AI agent is an LLM-based system formed by four core modules:*

- **Perception Module:** sensing the environment and converting external observations into internal representations.
- **Reasoning Module:** decomposing complex tasks, reasoning about dependencies, interacting with memory, and formulating execution strategies.
- **Memory System:** comprising short-term memory for immediate reasoning and long-term memory for experience retention to support the reasoning.
- **Action Module:** executing actions in the environments.

The objective of the AI agent is to maximize the desired evaluation metrics, e.g., accuracies, success rates, and rewards.

In this section, we present the preliminaries of AI agents. An AI agent is an LLM-based system to autonomously complete complex tasks by leveraging the LLM’s capabilities for reasoning, memorizing and decision making. The formal definition is presented in Definition II.1. A typical execution paradigm of AI agents is perception-reasoning-action cycle, where the AI agent receives the inputs from the environment

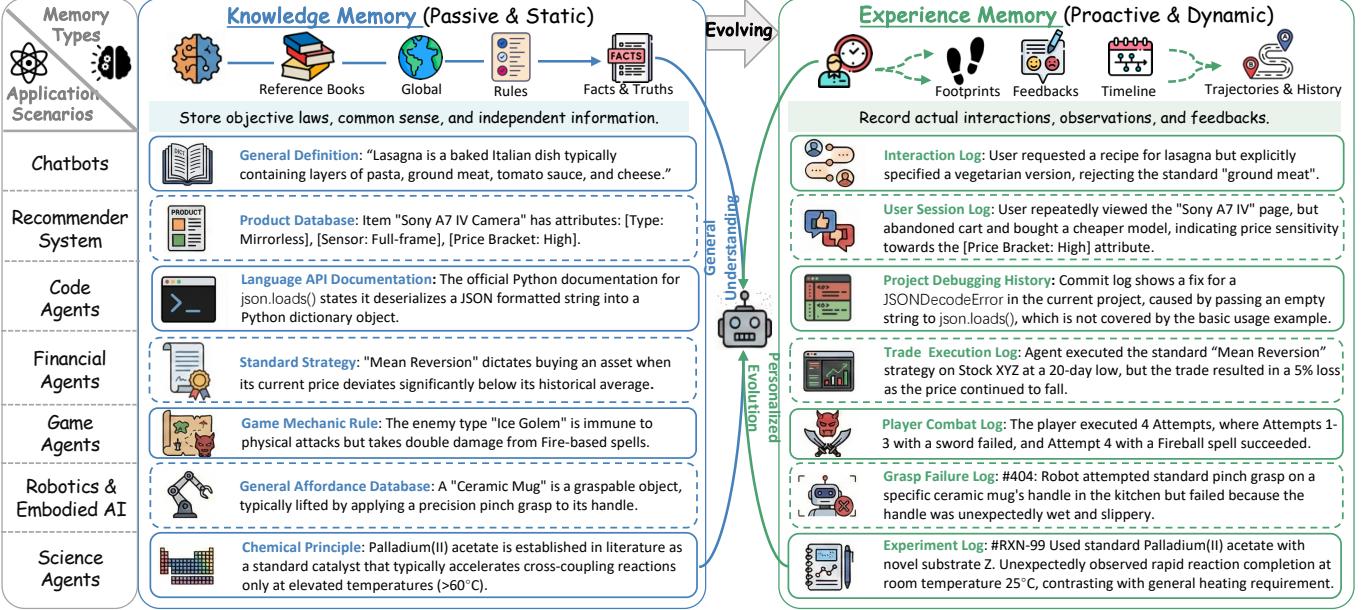


Fig. 2: Two types of agent memory, i.e., Knowledge Memory and Experience Memory, and their application across different agent scenarios. The synergy between static knowledge memory and dynamic experience memory enables agents to both understand the world's rules and adapt to personal interactions.

through its perception module, then reasoning with LLMs across the inputs, the LLM's internal knowledge and the contents stored in the memory system, and finally outputs the actions to execute into the environment. Within this paradigm, the memory systems play an important role for the AI agent's reasoning and action process. The basic operations of memory are defined in Definition II.2.

Definition II.2 (Basic Operations of Memory). *The basic operations of agent memory define the primitive actions that manipulate the memory \mathcal{M} . These atomic operations include:*

- **Write:** $\text{Write}(m, \mathcal{M}) \rightarrow \mathcal{M}'$, adding a new memory unit m into the memory repository \mathcal{M} .
- **Read:** $\text{Read}(q, \mathcal{M}) \rightarrow \mathcal{M}_q$, retrieving relevant memory units $\mathcal{M}_q \subseteq \mathcal{M}$ based on query q .
- **Update:** $\text{Update}(m, \mathcal{M}) \rightarrow \mathcal{M}'$, modifying existing memory units based on new information.
- **Delete:** $\text{Delete}(m, \mathcal{M}) \rightarrow \mathcal{M}'$, removing obsolete or irrelevant memory units from \mathcal{M} .

These operations collectively enable the dynamic management and evolution of the agent's memory system.

While the basic operations describe the atomic actions on memory, understanding how these operations are orchestrated over time is crucial for the full functionality of agent memory systems. Therefore, we further formalize the temporal dynamics of memory through the concept of lifecycle, which characterizes how information flows through different processing stages.

Definition II.3 (Lifecycle of Agent Memory). *The Lifecycle of Agent Memory describes the continuous, temporal flow of information processing within the agent, defining how raw data is transformed into stored knowledge and how the knowledge evolves over time. The lifecycle \mathcal{L} is defined as a cyclic process consisting of four distinct stages:*

- **Memory Extraction:** *The transformation of raw unstructured observations o_t into structured memory units m .*
- **Memory Storage:** *The organization and placement of extracted units into the memory structure \mathcal{M} through appropriate indexing and structural organization.*
- **Memory Retrieval:** *The mechanism of retrieving relevant stored information $\mathcal{M}_{\text{rel}} \subset \mathcal{M}$ in response to the query q .*
- **Memory Evolution:** *The post-processing phase where memory is refined, including internal self-evolving mechanisms (e.g., consolidation, abstraction) and external self-exploration processes (e.g., new environmental feedback).*

This lifecycle ensures that memory remains current, relevant, and optimally structured for supporting the agent's reasoning.

III. TAXONOMY OF AGENT MEMORY

In this section, we present a comprehensive taxonomy for agent memory from different perspectives. Specifically, memory is categorized on the basis of multiple dimensions, including temporal scope, functional roles and representational structures. We then introduce the graph-based memory as a unified view of agent memory and conclude this section from an implementation perspective.

A. Long-term vs. Short-term Memory

One of the most straightforward categorization of memory is the temporal dimension, i.e., information retention durations:

- **Short-term Memory:** Maintains recent, immediately relevant information with rapid access and limited capacity. This includes the current conversation context, active reasoning traces, and transient state variables. Short-term memory is volatile and typically discarded after the

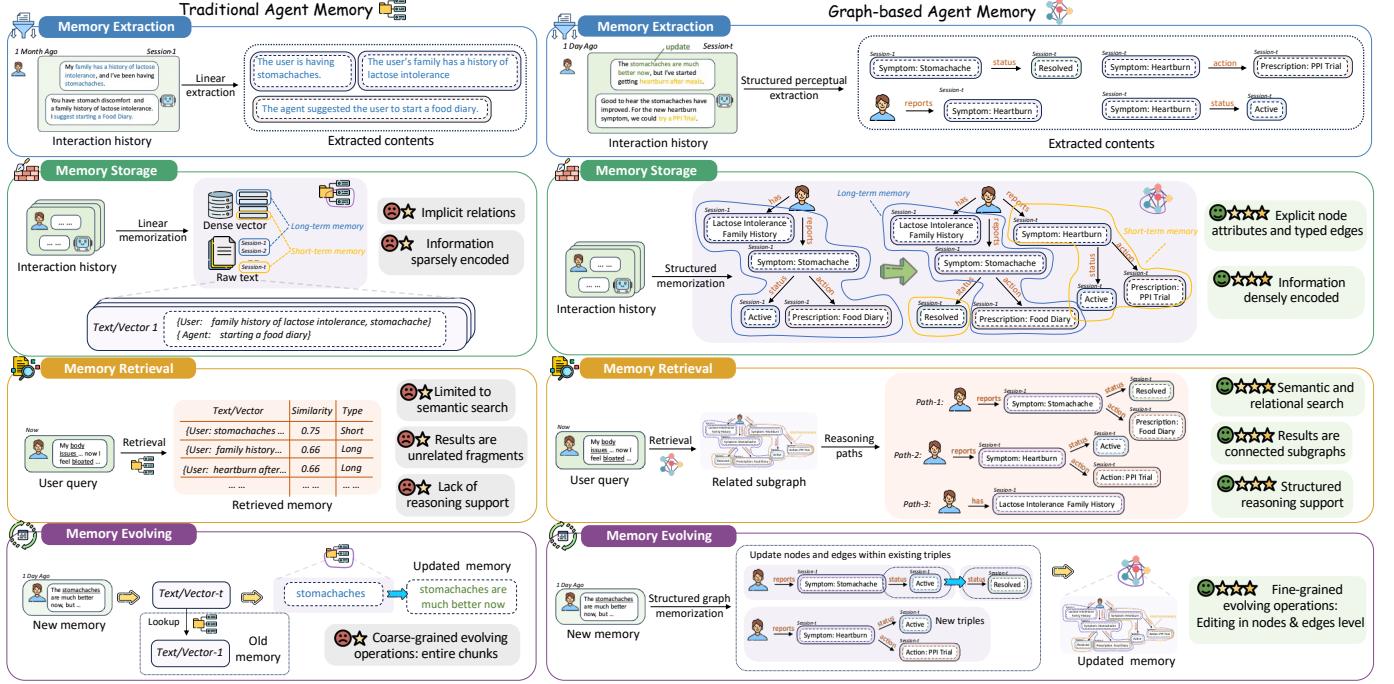


Fig. 3: Comparison between traditional agent memory and graph-based agent memory

immediate task completion, though significant elements may be consolidated into long-term storage.

- **Long-term Memory:** Stores persistent information across sessions, including accumulated knowledge, historical interactions, learned patterns, and user preferences. Long-term memory supports continuity across episodes, enables transfer learning, and provides the foundation for personalization and adaptive behavior over extended timescales.

B. Cognitive Structure of Memory

We introduce the cognitive structure of the memory in [8]:

- **Semantic Memory:** Stores general, decontextualized world knowledge and factual information (e.g., “Paris is the capital of France”). Within the graph structure, this forms the stable ontology, providing common sense and domain-specific facts that ground the agent’s reasoning.
- **Procedural Memory:** Encodes skills, routines, and immutable rules. It represents “how-to” knowledge, such as standard operating procedures or game rules. In application, this allows agents to execute complex tasks automatically under standard conditions.
- **Associative Memory:** Establishes latent links between different concepts or data points within the knowledge base. By connecting related pieces of information (e.g., symptoms to diagnoses, or products to categories), it facilitates creative reasoning and analogy-making, acting as the connective tissue of the knowledge graph.
- **Working Memory:** Acts as the agent’s “mental scratchpad” for immediate experience. It temporarily holds the current conversation turn, intermediate reasoning steps, and transient variables. While ephemeral, it is the entry point for all experience memory, characterized by fast access and direct influence on the immediate next action.

• **Episodic Memory:** Records the chronological sequence of past sessions. It transforms transient working memory into a persistent and queryable autobiographical history. For instance, it notes that a customer requested a vegetarian option or a canceled order. This allows the agent to recall “what happened and when,” providing temporal grounding.

• **Sentiment Memory:** Captures the emotional tone or sentiment derived from interactions. By logging user feedback or frustration levels, the agent can adapt its empathy and style in future turns. This layer adds a qualitative dimension to the raw interaction logs.

C. Knowledge vs. Experience Memory

To define agent memory concretely, we can draw a parallel to the human cognitive system. Human memory is a structured process involving the encoding, storage, and retrieval of information, often categorized into different types. Similarly, agent memory can be understood through two primary, complementary categories as shown in Figure 2:

Knowledge Memory (Passive & Static): This represents the agent’s passive and static repository of objective, global, and verifiable information. It functions as an internal reference library or textbook, containing canonical facts, universal rules, established procedures, and general truths about the world. This memory is typically pre-loaded, slowly updated, and context-independent. Its purpose is to provide a stable, reliable foundation for reasoning and action. For instance, it stores the factual definition of concepts, the immutable rules of a game, or standard scientific principles. In applications, a shopping agent’s product database or a robot’s general affordance model constitutes Knowledge Memory. Generally, this memory is passive, serving as a reliable, factual backbone for reasoning.

Experience Memory (Proactive & Dynamic): This is the agent’s personal logbook, actively recording its specific interactions, observations, and the outcomes of its actions. It includes user dialogue history, execution logs, trial-and-error trajectories, and feedback. For instance, it notes that a particular user requested a vegetarian lasagna variant, that a trade based on a mean-reversion strategy actually resulted in a loss, or that a standard grasping procedure failed on a wet mug. This memory is dynamic, personalized, and forms the basis for learning from practice and adapting to specific contexts.

D. Non-structural vs. Structural Memory

Traditional agent memory systems commonly adopt simple storage paradigms, including:

- **Linear or buffer-based memory**, such as fixed-length token windows or conversation histories, which maintain recent interactions but suffer from information loss and lack relational context.
- **Vector-based memory**, which encodes experiences into dense embeddings stored in vector databases, enabling semantic similarity search but struggling with structured reasoning and hierarchical relationships.
- **Key-value or log-based memory**, which records events in sequential logs or attribute-value pairs, supporting straightforward lookup but offering limited capability for complex querying or dynamic updates.

A common thread among these traditional paradigms is their treatment of memory as a sequential, flat, or implicitly structured store. While effective for certain patterns, they often fall short in explicitly representing and efficiently reasoning over the complex web of relationships between pieces of knowledge, a capability critical for sophisticated planning, causal understanding, and narrative coherence. More importantly, while these approaches enable basic recall and short-term context management, they exhibit notable limitations in scenarios requiring hierarchical knowledge organization, temporal tracking, and long-term adaptive learning. These constraints become particularly evident in long-horizon tasks, multi-session interactions, and domains where knowledge evolves dynamically.

E. Graph-based Memory: A Unified and General Perspective

Graph-based agent memory emerges as a powerful generalization and enhancement of conventional memory frameworks. The core idea of graph-based agent memory is modeling memory content as a dynamic, structured **Memory Graph**. In this paradigm, memory units (e.g., events, entities, concepts, observations) are abstracted as **nodes**, and the semantic, temporal, causal, or logical relationships between them are abstracted as **edges**. This explicit structural representation transforms memory from a flat list of entries or a hidden state vector into a rich, interconnected network of knowledge.

By representing memory elements as nodes and their relationships as edges, graph structures naturally support:

- ❶ Explicit relationship modeling, allowing agents to reason over causal dependencies and semantic associations between memory items;
- ❷ Hierarchical organization, from fine-grained

factual triples to general thematic clusters or subgraphs;

- ❸ Temporal and dynamic structuring, where time-aware edges can capture event sequences, state transitions, and knowledge evolution;
- ❹ Efficient structured retrieval, enabling traversal, subgraph extraction, and multi-hop relational queries beyond mere semantic similarity.

Notably, traditional memory forms can be viewed as degenerate or simplified cases within the graph memory paradigm. For instance, a linear buffer corresponds to a chain within a graph, and a vector memory can be interpreted as a fully-connected graph with similarity-weighted edges. Thus, graph-based memory does not merely replace existing designs but provides a unified and extensible framework.

In essence, graph-based agent memory elevates memory from a passive, flat “log” to an active, structured “knowledge graph” tailored to the agent’s lived experience. It not only records “what happened” but, more importantly, models “how these things are connected.” This provides a powerful foundation for associative reasoning, modeling long-term dependencies, and achieving explainable agent behavior. Figure 3 summarizes the key distinctions between traditional agent memory and graph-based agent memory. In summary, graph-based agent memory re-conceptualizes memory from a passive recording into an active, structured model of experience. By making relationships first-class citizens, it provides a powerful foundation for the complex reasoning, long-term coherence, and adaptive behavior required by sophisticated autonomous agents. Graph-based memory architectures, such as knowledge graphs and hypergraphs, have demonstrated superior performance in applications requiring multi-session coherence, personalized adaptation, complex task planning, and hallucination reduction. The subsequent sections delve into the technical realization of such memory systems, covering construction, retrieval, updating, and domain-specific applications.

F. An Alternative View from Implementation

In this paper, we use the lifecycle to introduce the memory, however, we can also have an alternative view of the memory from the implementation perspective, which can help readers to understand this paper as a researcher or engineer. Specifically, the framework begins with underlying structures, which encompasses the foundational data representations including graph-based structures, embeddings, and temporal sequences that form the basis of memory storage (Section V). The stored representations manifest as various memory contents, including conversational history, temporal records, and structured knowledge bases. These contents are curated by memory processes of extraction (Section IV) and evolution (Section VII), where relevant information is filtered and refined based on context and usage patterns. Finally, the usage interface enables Retrieval (Section VI) of relevant information and facilitates reasoning tasks by accessing this curated memory. This implementation view provides a systematic perspective from the implementation on how language models maintain, process, and leverage memory to support the complex reasoning.

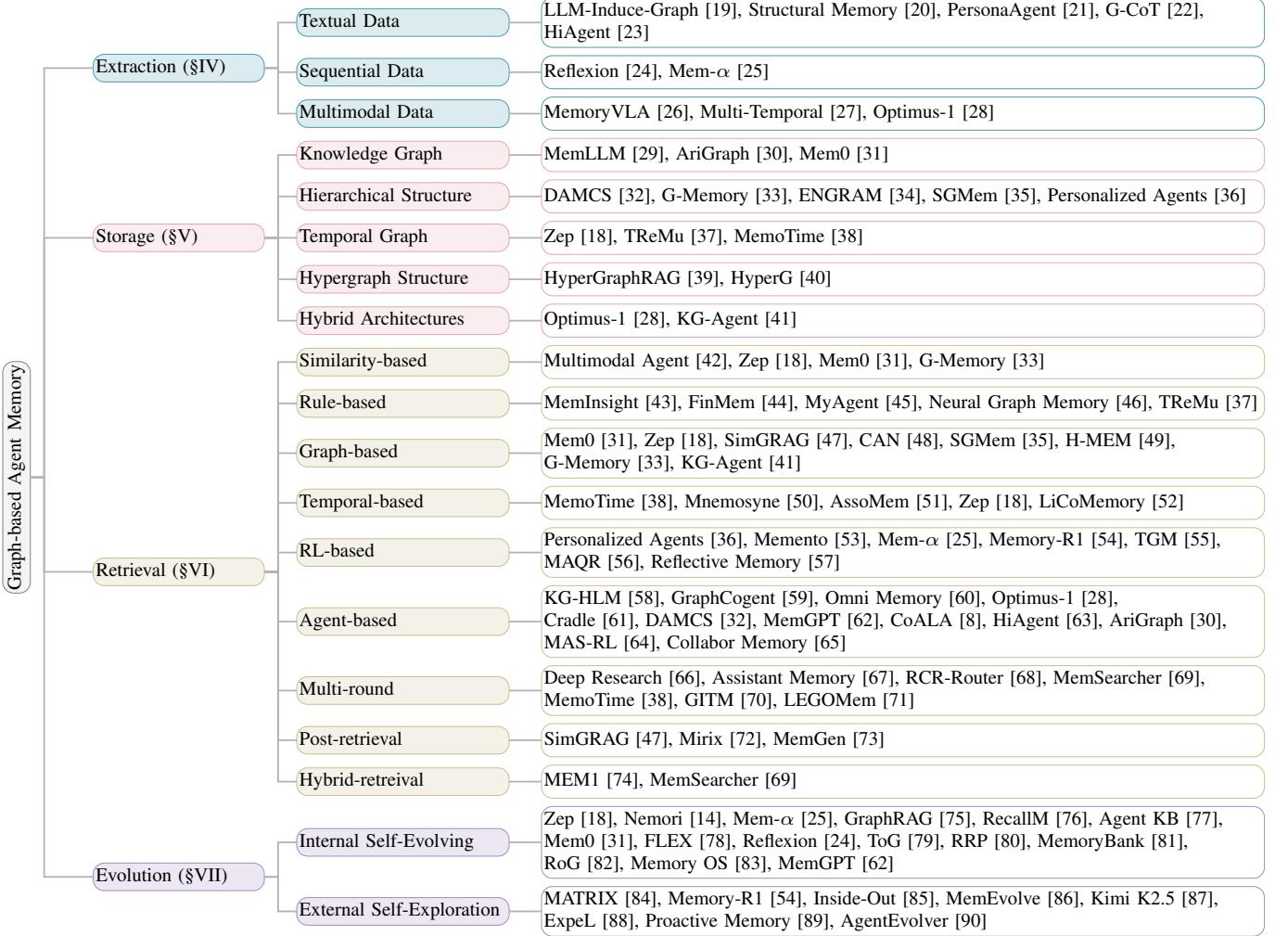


Fig. 4: A Comprehensive Taxonomy of Graph-based Memory Management for LLM Agents.

IV. MEMORY EXTRACTION: TRANSFORMING THE DATA

The process of memory extraction begins with the collection and pre-processing of raw data. These raw inputs, which vary widely in format and constitute the foundational material from which an agent's internal memories are constructed. From this perspective, memory extraction can be understood along two complementary dimensions: the source from which memory is derived and the form in which the extracted memory is ultimately encoded. Based on the definitions introduced in the previous section, agent memory can be broadly categorized into knowledge and experience memory, which differ primarily in their sources and roles within the system.

Experience memory is extracted from the agent's own interaction history and reflects the accumulation of situated, task-specific experience over time. Its primary sources include multi-turn dialogues with users, action and observation sequences produced during task execution, and explicit or implicit feedback signals. **Knowledge memory**, in contrast, is extracted from sources that exist independently of the agent's personal experience and are intended to represent objective and generally valid information. Typical sources include curated knowledge bases, domain-specific databases,

formal documentation, instructional texts, and other authoritative corpora. In addition, a substantial portion of knowledge memory is implicitly acquired through pretraining on large-scale textual data, where factual and procedural information becomes embedded in the model parameters. Information drawn from these sources is usually stable, context-independent, and broadly applicable. Consequently, knowledge memory extraction focuses on distilling canonical facts, rules, and procedures into persistent representations that provide a reliable foundation for reasoning across tasks and domains.

The underlying information shows up in many different formats. These sources may appear as unstructured or semi-structured text such as documents, dialogue transcripts, and execution logs. Others may be non-textual, including images, sensory observations, or structured records generated during interaction. This diversity in source modality means that raw information cannot be directly stored as memory. Instead, it must be processed and abstracted in ways that reflect both the characteristics of the source and the type of memory being constructed. As a result, different sources naturally require different extraction strategies. The following outlines the primary techniques used for different types of data sources.

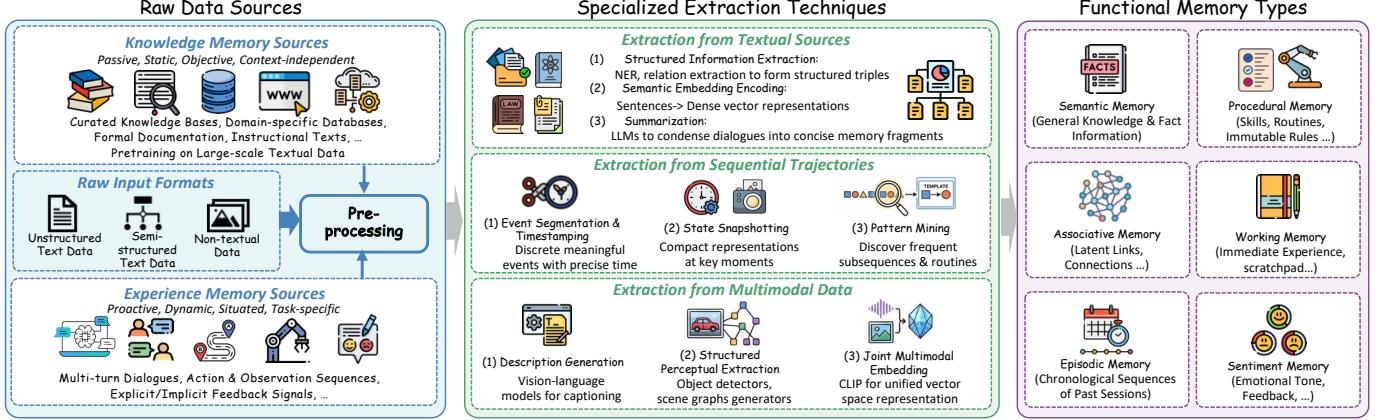


Fig. 5: Overview of agent memory extraction. This figure illustrates a unified pipeline of agent memory construction from various data resources. Raw inputs, originating from both experience and knowledge memory, are transformed into structured and compact representations through specialized extraction techniques. These extracted units are then organized into distinct functional memory types, enabling agents to support reasoning and downstream tasks.

a) *Extraction from Textual Sources*: The extraction of memory from unstructured textual data, such as conversational logs or documents, focuses on identifying and structuring semantic information [19]. Key approaches include: (1) Structured information extraction: Using named entity recognition and relation extraction models, or prompting large language models, to directly extract entities, attributes, and their relationships in the form of (entity-relation-entity) triples [20], [21]. (2) Semantic embedding encoding: Encoding sentences or paragraphs into dense vector representations using models such as Sentence-BERT, transforming semantic content into embeddings suitable for similarity-based retrieval [22]. (3) Summarization: Applying extractive or abstractive summarization models, including large language models, to condense lengthy texts or dialogue histories into concise and informative memory fragments [23].

b) *Extraction from Sequential Trajectories*: For time-series interaction data, such as action-observation sequences, extraction aims to capture temporal structure: (1) Event segmentation and timestamping: Segmenting continuous trajectories into discrete, meaningful events or episodes and annotating each with precise timestamps [24]. (2) Dynamic State Snapshots: Beyond static snapshots, this method captures the evolution of critical states. It involves periodically capturing and storing compact representations of the agent's or environment's state at key moments, such as embeddings or feature vectors [25]. (3) Pattern mining: Applying offline sequence mining or clustering algorithms to discover frequently occurring successive subsequences or strategic routines, which are then abstracted into procedural memory templates.

c) *Extraction from Multimodal Data*: For sensory data such as vision or audio, extraction bridges the gap between raw signals and semantic meaning: (1) Description generation: Using vision-language models [26] or audio captioning models [27] to generate textual descriptions of visual scenes or auditory events, which are subsequently processed as text. (2) Interactive content: This process focuses on detecting objects or distilling information between an agent's actions and environmental responses such as raw pixels or signals.

(3) Joint multimodal embedding: Encoding data from different modalities into a unified vector space using customized models, producing a joint embedding that serves as a compact memory representation of the multimodal experience [28].

As shown in Figure 5, the raw data is transformed into structured and semantically enriched representations by the extraction process. This transformation typically occurs across three levels of abstraction: from the original raw data flows, to intermediary extracted entities and relationships, and finally into organized functional memory types that serve specific cognitive and operational purposes within the agent, which is introduced in Section III.

V. MEMORY STORAGE: ORGANIZING THE MIND

The extraction stage described above produces a set of semantically enriched artifacts, including identified entities and relations, dense semantic embeddings, concise summaries, segmented events with timestamps, and multimodal captions, which together form the operative substrate for downstream memory architectures. A central challenge in building agent memories is therefore to transform these heterogeneous extracted artifacts into storage formats that preserve relevant semantics while supporting efficient retrieval and reliable updates. Unlike static knowledge bases, agent memories must also accommodate dynamics, personalization and experience grounding, all of which shape how extracted information should be encoded and maintained.

Choosing a particular graph based memory structure amounts to explicit trade offs between competing design goals. Precision and explicit multiple hop reasoning favor relational graphs; compression and conceptual abstraction favor hierarchical or treelike summaries; temporal fidelity motivates temporal knowledge graphs and time indexed episodes; and cross modal generalization or fuzzy recall often favors vector stores or hybrid systems. With these trade offs in mind, this section focuses first on Knowledge Graphs as a canonical relational substrate, explaining how triples are produced, integrated and

maintained, and then surveys hierarchical, temporal, hypergraph and hybrid architectures that address complementary points along this design spectrum. Figure 6 summarizes the construction paradigms discussed below.

A. Knowledge Graph Structure

The Knowledge Graph (KG) stands as a structured memory paradigm, explicitly designed to store and reason over factual knowledge [29]. It represents information as a network of interconnected *triples*, where each triple takes the form (*head entity, relation, tail entity*). This relational structuring makes it a powerful substrate for specific types of agent memory.

a) KG Modeling: Constructing a KG for an agent involves the continuous extraction and integration of structured triples from unstructured interaction streams. The primary method relies on Large Language Models serving as a powerful, open-vocabulary parsing engine. For instance, in the AriGraph world model [30], the LLM parses each textual observation from the environment to identify relevant objects and extract their relationships in the form of triplets. Similarly, systems like Mem0 [31] employ an LLM in their extraction phase to convert conversation messages into entity-relation triplets. This approach outperforms traditional Named Entity Recognition (NER) and relation classification pipelines, as LLMs can generalize to novel fine-grained entity types and exploit implicit relations based on contextual understanding.

The extracted triples then enter an update phase, where they are integrated into the persistent graph store. This phase is non-trivial, involving operations such as conflict detection (e.g., handling contradictory facts), relationship pruning, and schema evolution. Mem0 explicitly models this through a dedicated update mechanism that evaluates new memories against similar existing ones. This two-step process, LLM-based extraction followed by reasoned integration, forms the core pipeline for building a dynamic, experience-driven knowledge graph from an agent’s raw perceptions.

b) Corresponding Memory Types: The triple-based structure of a KG makes it exceptionally suitable for implementing long-term static memory types. This includes general world facts (e.g., *(Paris, Capital_Of, France)*) and immutable domain-specific concepts. The explicit relational format allows for efficient storage and complex, multi-hop queries that are difficult for vector-based retrievers. Furthermore, by augmenting triples with temporal metadata or linking them to episodic events, where episodic vertices are connected to triples extracted from the same observation, KGs can also support episodic memory.

B. Hierarchical Memory Structure

The hierarchical structure is the most common and intuitive paradigm for organizing knowledge within agent memory systems [32], [33], [36]. By arranging information into multi-level trees, it provides a rationale hierarchy to compress extensive experiences into manageable schema. The essence of a tree is a directed acyclic graph (DAG) that explicitly models *parent-child* and *containment* relationships, allowing it to represent concepts ranging from broad categories to specific instances and from high-level goals to granular execution steps.

This inherent property enables efficient top-down navigation for querying abstract themes and bottom-up summarization for maintaining coherence across vast memory stores. Systems like MemTree adopt this approach by dynamically routing new information through the hierarchy, clustering similar content under existing nodes while creating new branches for novel information, with all ancestor nodes recursively updating their semantic summaries to reflect the integrated knowledge.

The construction of hierarchical memory mainly relies on two processes: **semantic clustering** for organization and **recursive summarization** for abstraction [34]. To maintain the hierarchy’s value as a compressed knowledge schema, each parent node dynamically synthesizes a concise linguistic summary of all information contained within its subtree. Beyond static factual knowledge mentioned above, such structure also manifests as *Session or Sequence Memory*. Here, individual sentences or dialogue turns, are linked primarily by their temporal order or conversational flow within a defined session. This creates a timeline or narrative chain that is essential for modeling dialogue coherence and user intent evolution over a single interaction episode. The SGMEM [35] system proposes a sentence-level tree for long-term conversational agents, using the sequential and referential links between utterances to construct a coherent interaction.

C. Temporal Graph Structure

Real-world interactions are inherently dynamic, where facts hold validity only within specific time windows. Temporal Knowledge Graphs (TKGs) extend standard triples to quadruples (s, r, o, t) , but simple timestamping is often insufficient for complex agentic workflows. Recent architectures have introduced more granular temporal modeling to handle validity, ambiguity, and reasoning monotonicity.

a) Bi-Temporal Modeling: A critical distinction in agent memory is between the time an event occurs (*valid time*) and the time it is recorded (*transaction time*). **Graphiti** [18] implements a bi-temporal model tracking two distinct timelines. This allows the system to manage evolving conversations where a fact introduced at t_1 might retrospectively describe an event at t_2 . By explicitly tracking creation ($t'_{created}$) and expiration ($t'_{expired}$) timestamps alongside validity intervals, the system can resolve contradictions through temporal invalidation rather than overwrites, maintaining a faithful history of state changes.

b) Disentangling Mention and Event Time: In multi-session dialogues, relative temporal expressions (e.g., “last Friday”) often create ambiguity. **TReMu** [37] employs a time-aware memorization mechanism to decouple the *mention time* (session timestamp) from the inferred *event time*. TReMu structures memory as “timeline summaries,” where events are grouped and indexed by their inferred absolute timesteps. This structure supports neuro-symbolic reasoning, where the agent generates Python code to perform precise arithmetic on dates before retrieving the corresponding timeline nodes.

c) Hierarchical Temporal Constraints: **MemoTime** [38] is developed to prevent logical hallucinations in reasoning chains (e.g., retrieving an effect that occurred before its cause). This framework organizes the TKG reasoning process into a

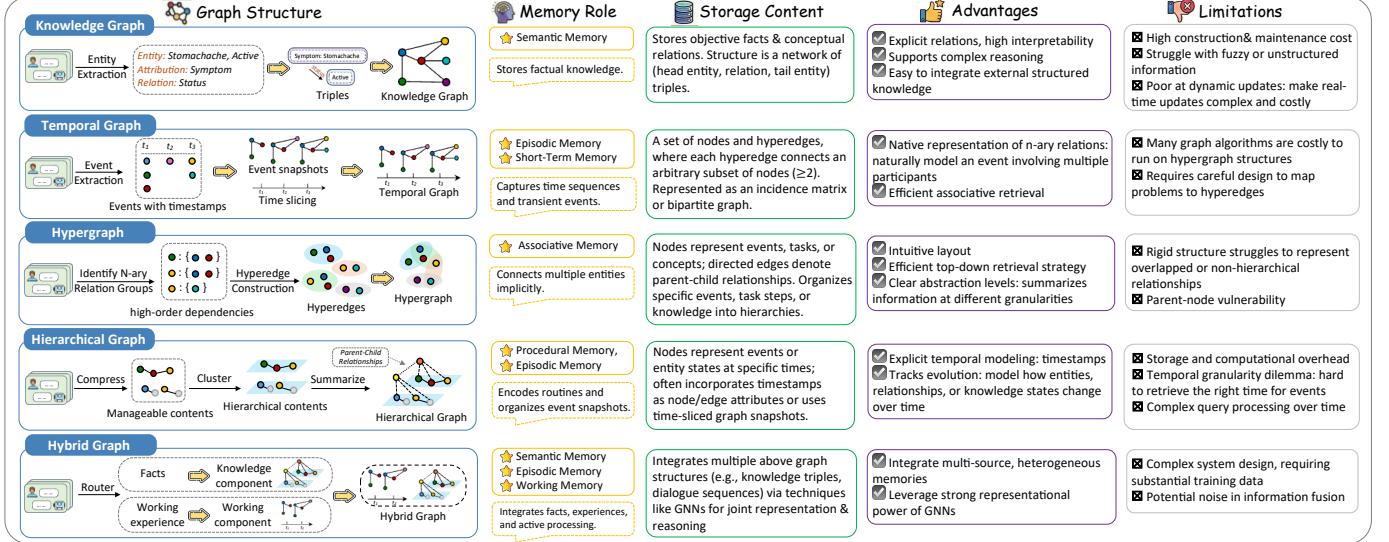


Fig. 6: A Comprehensive Taxonomy of Graph Construction Paradigms for Agent Memory Systems: Methodologies, Corresponding Memory Functions, and Comparative Analysis of Advantages and Limitations.

hierarchical “Tree of Time.” Unlike flat retrieval, this structure enforces *temporal monotonicity*, ensuring that any retrieved reasoning path $e_1 \rightarrow e_2 \rightarrow e_3$ strictly adheres to chronological constraints ($t_1 \leq t_2 \leq t_3$). This operator-aware design allows the agent to prune semantically relevant but chronologically invalid evidence effectively.

D. Hypergraph Structure

While binary graphs (connecting two entities) are efficient, they suffer from information loss when representing complex, multi-entity interactions (e.g., three drugs acting together to cause a side effect). Hypergraphs address this by employing hyperedges that can connect an arbitrary number of nodes, preserving the integrity of n -ary relations.

a) *Information Integrity in N-ary Relations:* The primary motivation for hypergraph memory is to prevent the sparsity and semantic fragmentation caused by decomposing complex facts into binary edges. **HyperGraphRAG** [39] demonstrates that hypergraph-structured representation is information-theoretically more comprehensive than binary equivalents. By treating a natural language knowledge fragment and all its associated entities as a single hyperedge $e_i = (e_i^{text}, \{v_1, \dots, v_n\})$, the system enables “dual-retrieval”, simultaneously retrieving relevant entities and the hyperedges that bind them, thereby accessing complete facts for generation.

b) *Structural Dependency in Tabular Data:* Hypergraphs are particularly effective for structured data where relationships are inherently group-wise. **HyperG** [40] models tabular knowledge using hypergraphs. It constructs distinct hyperedges for rows, columns, and the entire table to capture high-order dependencies, such as semantic consistency within columns and the hierarchical relationship between captions and cells. To enhance reasoning, HyperG utilizes a Prompt-Attentive Hypergraph Learning (PHL) module, which dynamically propagates attention between nodes and hyperedges based on the specific

inquiry, effectively simulating a human-like focus on relevant data substructures.

E. Hybrid Graph Architectures

Graph structures excel at precision and multi-hop reasoning but may lack the breadth of vector retrieval or the flexibility of unstructured buffers. Hybrid architectures fuse graphs with other data structures to balance these trade-offs, typically separating “static knowledge” from “dynamic experience.”

a) *Knowledge-Experience Decoupling:* A leading design pattern is to separate world rules from agent trajectories. *Optimus-1* [28] proposed a **Hybrid Multimodal Memory** for agent. It combines a *Hierarchical Directed Knowledge Graph (HDKG)* to store static, structured game mechanics (modeled as directed acyclic graphs for crafting recipes) with an *Abstracted Multimodal Experience Pool (AMEP)*. The AMEP functions as a dynamic vector store that retains multimodal success and failure trajectories. This separation allows the agent to ground planning in rigid graph-based knowledge while refining execution by retrieval-augmented experience from the pool.

b) *External Graph with Internal Working Memory:* Another hybrid approach focuses on the interaction between an external massive graph and an internal lightweight state. The **KG-Agent** framework [41] integrates an external knowledge graph with an internal *Knowledge Memory*. Unlike purely graph-based agents, KG-Agent maintains a structured scratchpad that iteratively updates the reasoning history, tool definitions, and intermediate observations retrieved from the external KG. This hybrid design enables the agent to navigate large-scale static graphs using a flexible, evolving internal context, bridging symbolic storage with neural reasoning.

In summary, different types of memory often require different approaches to graph construction, and there is no single paradigm that fits all scenarios. Knowledge memory, which is generally stable, structured, and context-independent, is well suited to graphs that emphasize relational or containment

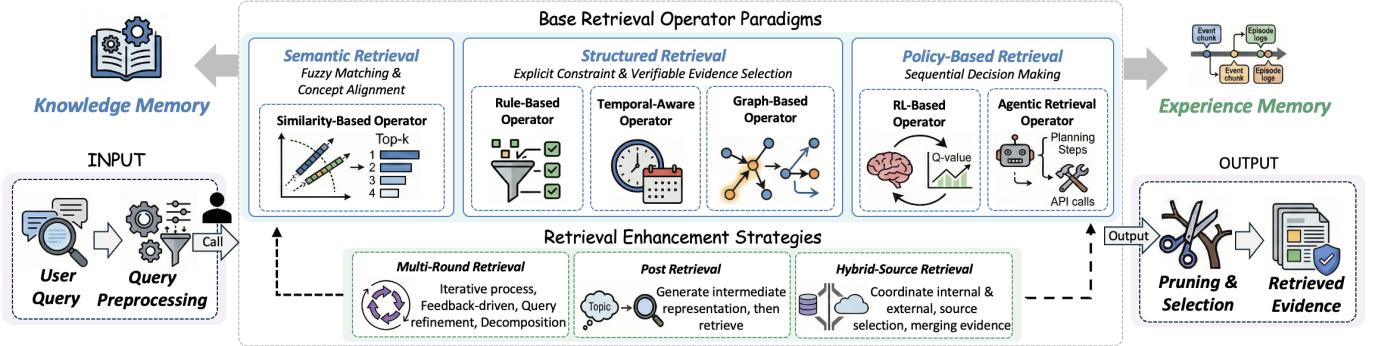


Fig. 7: Retrieval pipeline architecture integrating base operators and enhancement strategies. **Left:** User queries go through preprocessing before retrieval. **Top:** Six base retrieval operators are organized into three paradigms: semantic, structured, and policy-based retrieval, which interact with knowledge and experience memory types. **Bottom:** Retrieval enhancement strategies layer on top of these operators: multi-round retrieval, post-retrieval, and hybrid-source retrieval coordinates internal memory with external resources. **Right:** Final pruning and selection produce ranked retrieved evidence for downstream reasoning with these operators and strategies.

structure, such as knowledge graphs. These graphs can capture explicit relationships between factual entities, and enable efficient reasoning over globally valid information. In contrast, experience memory is dynamic, personalized, and context-specific, and often benefits from graphs that emphasize temporal sequences, interaction trajectories, or user-action networks. Such graphs capture the evolving patterns of agent interactions or preferences over time. The choice of graph type is therefore closely linked to the properties of the underlying memory. Factors such as sparsity, temporal dynamics, modality diversity, and the need for incremental updates all influence how nodes and edges are defined. Moreover, hybrid or multi-layered graphs are sometimes necessary to integrate both knowledge and experience memory. Overall, constructing graphs requires careful consideration of the characteristic of memory types.

VI. MEMORY RETRIEVAL: RECALLING THE PAST

These construction choices directly shape the memory usage. Once memory structures are fixed, the system must decide how to access them to support downstream reasoning. This motivates the retrieval stage, the next stage of the storage pipeline. The retrieval stage manipulates graph memory by defining executable operators. Base retrieval operators can be organized into three paradigms. These paradigms interact with the two memory roles: knowledge and experience.

Semantic retrieval including similarity-based operators operates over extracted text or multimodal chunks and their embeddings. It supports fuzzy matching and basic concept alignment. It is often used as a candidate generator for knowledge and experience memory.

Structured retrieval includes rule-based operators, temporal operators and graph-based operators. These operators execute explicit constraints over structured artifacts: knowledge graphs, hierarchies, temporal graphs, and hypergraphs. This enables verifiable and interpretable evidence selection: each retrieval decision is traceable through the underlying graph structure or rule set. Structured retrieval is particularly central to knowledge

memory. For knowledge memory, correctness and consistency are primary concerns.

Policy-based retrieval includes RL and agentic retrieval operators that treat retrieval as sequential decision-making. The system selects which memory type to query and chooses which operators to apply. It allocates computational budgets and decides when to stop retrieval. Policy-based retrieval is especially important for experience memory, which is dynamic, personalized, and time-sensitive in nature.

In practice, systems commonly combine these basic operators, e.g., semantic anchoring → structured expansion → policy-controlled stopping and pruning.

A. Categorization of Retrieval Techniques

The retrieval process can be broken down into three basic operations: (i) query preprocessing, (ii) candidate retrieval, and (iii) pruning. They are usually executed as a simple pipeline that extracts a small set of relevant evidence from the memory graph. In this section, we introduce a set of base retrieval operators that can be flexibly composed to implement the latter two operations. Together with preprocessing, these operators form an end-to-end retrieval pipeline. Beyond these operators, we also describe retrieval enhancement strategies, which are auxiliary practices layered on top of the base operators to improve retrieval quality.

1) *Similarity-based Operator*: Similarity-based retrieval is a coarse retrieval operator to encode a user query into a vector and then retrieves the top- k most similar memory entries in the embedding space [42].

For knowledge memory, similarity-based operator primarily supports exact abstract concept or entity recall. For experience memory, the operator functions as specific memory unit recall, e.g., episodes, summaries, or agent states which match the current query. In practice, systems often combine similarity retrieval with experience memory and knowledge memory [31]. For experience memory, systems use summary graphs where the retriever first matches high-level summaries and then drills down to supporting raw corpus chunks. For knowledge memory,

systems use knowledge graphs where retrieved entities or triples serve as anchors for subsequent graph expansion.

In simpler applications, direct similarity search can be effective [18]. It can also play a supporting role. For instance, it can use query-query [33] similarity to filter unrelated memories. However, this approach has clear limitations:

- **Similarity does not guarantee relevance.** Lexically or semantically similar text may not match the specific memory needed for a task.
- **Poor multi-hop reasoning.** Answers to complex queries often depend on entities not in the original query. Pure similarity matching cannot bridge this gap.
- **Temporal awareness is often missing.** Memory retrieval in dynamic contexts requires understanding time and situational relevance, not just semantic proximity.
- **Scalability introduces noise.** As memory grows, the number of semantically similar segments increases. This leads to redundant or irrelevant retrievals, which degrade precision. These challenges motivate more sophisticated retrieval beyond similarity search. Practical systems should leverage structured retrieval that align with the organization of memory. Moreover, retrieval should be policy-driven: an agent selects and composes operators conditioned on the underlying memory type.

2) *Rule-based Operator*: Rule-based operator in graph memory uses symbolic rules, and executable filters to decide relevant memories rather than purely semantic similarity.

For knowledge memory, rule-based operator serves two roles. It can act as a first-stage selector that preprocesses candidates for downstream retrieval, or as a post-retrieval validator that prunes retrieved memory by enforcing hard constraints. For experience memory, rule-based retrieval primarily supports temporal scoping over dynamic episodes such as dialogue histories and execution trajectories, since experiences are noisy and continually evolving [45]. In practice, for knowledge memory, rule-based operators can restrict candidates to compatible entity or relation types, prioritize high-confidence triples, and exclude triples flagged as conflicting by predefined rules. For experience memory, they leverage common rules such as time-window filters and task-phase constraints like execution [43].

A common variant uses hand-crafted associative heuristics inspired by Hebbian-style updates [46]: memories that are frequently co-retrieved may have their links strengthened, and recently written or repeatedly referenced items can be up-weighted over time. Such lightweight update rules can improve coherence by retrieving bundles of mutually associated memories rather than isolated items [44].

Rule-based retrieval also frequently applies deterministic symbolic filtering to satisfy explicit query constraints. In structured backends, e.g., SQL databases, these constraints can be compiled into executable queries or programs, e.g., SQL or Python code that queries and joins tables, improving precision and making the retrieval pipeline auditable [37].

3) *Temporal-Based Operator*: Temporal-based operator is essential for handling queries that depend on when events occurred, tracking facts that change over time, and preserving the sequence of interactions in conversations [38], [50].

For knowledge memory, general truths remain valid regardless of when they were learned, so temporal operators

primarily serve as a filtering rule rather than as a core retrieval mechanism. For experience memory, which records interactions and events whose relevance shifts with time, temporal operators become primary. A user's interests evolve, recent tool failures matter more than old ones, and the system must distinguish between what worked last week versus last year. Temporal operators therefore rank episodes by how recently they occurred, downweight outdated preferences as time passes, and retrieve chains of events that form coherent narratives rather than isolated records from different time periods.

In practice, Zep [18] maintains explicit time windows for each fact, marking when it becomes valid and when it expires. This enables the system to determine whether a fact still applies at query time. LiCoMemory [52] applies a decay function during ranking that reduces the weight of older facts while preserving highly relevant information. Systems also parse the user's query to infer the intended time range, such as extracting "last month" from "What did I discuss last month?", then limit retrieval to that window [51]. These approaches improve recall for questions that inherently depend on timing.

4) *Graph-based Operator*: Graph-based retrieval operator traverses explicit relational links in experience or knowledge graph memories. It uses query-conditioned traversal to expand from anchor nodes into a task-relevant subgraph in the same graph or another graph.

For knowledge memory, a graph makes reasoning support explicit by encoding facts as linked units. Retrieval can enforce structured relational constraints and return justificatory structures such as short paths or induced subgraphs. It enables compositional queries that join multiple facts across the graph [41]. For experience memory, a graph supports reasoning by preserving the connectivity of episodes. It lets the agent reconstruct coherent state-action-reward chains and attribute rewards to specific conditions. It retrieves adjacent follow-up actions that are central to proactive adaptation [35].

In practice, knowledge and experience memory share a common working scenario: first, identify key entities as anchors; second, expand candidates via neighborhood traversal, e.g., within n hops; third, score and prune nodes or paths to obtain compact evidence. In knowledge memory, anchors are typically entities or concepts, and traversal is often relation-constrained over a knowledge graph to support logic-like retrieval. In experience memory, anchors are commonly situation descriptors extracted from dialogue chunks and execution logs, such as tool outputs, or environment states, and traversal expands along event and temporal links to recover episodic context [47].

a) *Intra-layer Traversal*: For instance, Mem0 [31] follows an entity-centric approach that expands relations around identified entities. Zep [18] augments retrieval with breadth-first neighborhood expansion to collect candidate nodes or edges within a bounded hop radius, which can then be re-ranked by relevance. H-MEM [49] uses index-based routing on a hierarchical memory tree to retrieve relevant content layer by layer. Some approaches further employ GNN-based models to obtain cross-layer representations for retrieval and scoring [48].

b) *Inter-layer Traversal*: In hierarchical graph memories, traversal also operates across abstraction levels via inter-layer edges, e.g., edges between summary nodes and dialogue chunks,

enabling bottom-up abstraction. G-Memory [33] performs bi-directional traversal that synthesizes generalized strategies from an insight graph with detailed logs from an interaction graph. LiCoMemory [52] uses explicit hyperlinks to traverse from abstract summaries to precise dialogue chunks containing supporting evidence. Trainable graph memory [55] aggregates cross-layer path strengths across its query, transition path, and meta-cognition layers to derive relevance scores for retrieval.

5) *Reinforcement Learning-based Operator*: Reinforcement learning (RL) is increasingly used as a training operator to learn adaptive retrieval policies by optimizing downstream task rewards defined by the underlying memories.

The two memory types serve different purposes: knowledge memory mainly provides stable grounding and constraint for this operator, whereas experience memory supports context-sensitive adaptation. Context-sensitive adaptation means adapting retrieval action to the agent's current context, such as the current task state, by recalling and ranking previously observed state-action-reward episodes that match the current state. They share the same purpose: to retrieve evidence that is consistently helpful while down-weighting evidence candidates that are semantically similar yet systematically misleading [25]. This improves robustness beyond fixed similarity heuristics. In practice, RL-based retrieval operator can be applied to any structured memory systems whenever retrieval actions are well-defined, a reward signal reflecting retrieval utility is available, and the policy can be trained to balance performance gains against retrieval cost [56].

A common instantiation augments embedding-based candidate generation with a learned action-value function $Q(s, a)$, where the state s summarizes the current query and optionally the agent state. The action a corresponds to a retrieval decision such as selecting memory items, choosing a graph-navigation step, or issuing a tool call [36], [53].

Beyond single-step selection, some works formulate retrieval as a sequential decision process and train a dedicated memory agent with policy optimization, e.g., PPO/GRPO [55]. In a typical pipeline, the memory agent decides what to retrieve and in some designs also what to write into memory. It then passes the retrieved memory M_{ret} to a possibly frozen answer agent to produce the final response. Rewards are computed from task-specific answer quality metrics, e.g., EM/F1 for QA, and used to update the retrieval policy to maximize performance [54].

Since end-to-end online RL over LLMs can be expensive, an alternative is RL-free policy learning via retrospective labeling. In such schemes, an expert LLM evaluates which retrieved memories are genuinely useful, producing supervision signals to train a lightweight retriever, e.g., an MLP ranker. Another way is to use lightweight heuristic scorers to guide selection when the candidate set is large [54], [64]. This offers a pragmatic trade-off between adaptivity and training cost when large-scale online RL is impractical [57].

6) *Agent-based Operator*: Agent-based operator treats retrieval as an open planning-feedback loop where the agent can step beyond the boundaries of internal memory graphs [30], [58]. The core distinction from RL-based approaches lies in the ability to call external tools and APIs to supplement internal memory, then store verified results back into memory for future

use [65]. This enables retrieval to extend beyond selecting from a fixed knowledge base.

For knowledge memory, the agent traverses the knowledge graph to gather supporting facts via self-planning [59]. Graph traversal allows the agent to discover chains of reasoning within the structured memory. When internal graph coverage is incomplete, the agent can leverage internal knowledge to justify information correctness. For experience memory, the agent does not traverse a graph but instead traces chains of actions across episodes to recover episodic context [8], [63]. Rather than navigating structured relationships, the agent reconstructs events from timestamped logs to make decisions [47].

The action space includes selecting the corresponding store and index and choosing between a knowledge-graph index for rules or facts and a temporal or hypergraph index for past episodes [61]. The agent navigates the selected memory structure by picking the next node or edge to visit or moving across abstraction layers. The agent maintains a bounded working memory that records the current question, the partial plan, which memory source each piece of evidence comes from, previously visited nodes, and the evidence collected so far. This enables closed-loop navigation decisions [28], [60].

Beyond pre-defined action spaces, the agent can write API calls, e.g., SQL queries or search requests, to retrieve information on-demand actively. This aligns retrieval decisions with the agent's planning process [32], [62]. In complex systems, responsibilities can be split across multiple agents, e.g., candidate proposal vs. ranking, to improve robustness.

B. Retrieval Enhancement Strategies

Beyond base retrieval operators, recent graph memory systems increasingly emphasize retrieval enhancement strategies. These methods do not define a new operator by themselves. Instead, they improve retrieval by adding extra steps around a base operator rather than one-shot look up. These strategies also help in both the two memory roles discussed earlier. Knowledge memory requires enhancement strategies that prioritize reliability and conflict handling, since stable facts and rules must remain consistent across retrievals. Experience memory requires enhancement strategies that prioritize temporal coherence, personalization, and iterative evidence gathering, since it captures what actually happened to the agent and user.

In this survey, we group them into three classes: i) **multi-round retrieval**, which increases search depth and coverage via repeated retrieval; ii) **post-retrieval**, which makes the query clearer by first generating an intermediate representation, e.g., topic or intent description and then retrieving; and iii) **hybrid-source retrieval**, which improves answer completeness by retrieving from both internal memory and external sources and then combining the results.

1) *Multi-round Retrieval*: Multi-round retrieval treats memory access as an iterative process rather than a single-pass query over the memory [66]. Each round generates the next retrieval query conditioned on the original query and previously retrieved memory, retrieves additional memories, and then aggregates and evaluates whether the accumulated evidence is sufficient [68], [69]. If sufficient, it terminates; otherwise, it triggers another

round. More broadly, multi-round retrieval can be implemented with policies that decide whether to re-query, how to refine queries, and when to stop. This loop makes retrieval explicitly feedback-driven rather than static [66], [67].

Moreover, some systems decompose a complex query into sub-queries [38] and perform retrieval at the sub-query level, optionally rewriting each sub-query to make the queries more expressive. This fine-grained decomposition can reduce semantic drift of query and enable targeted evidence gathering [15].

For knowledge memory, extra rounds are often used to collect supporting facts or rules and to verify consistency across retrieved items. For experience memory, extra rounds are often used to reduce missing context, such as recent user requests, recent failures, or the latest environment changes. In practice, this also changes the stopping rule: knowledge-oriented loops stop when evidence is consistent and well supported, while experience-oriented loops stop when enough recent and relevant episodes have been gathered [70], [71].

2) *Post-retrieval*: While most memory-augmented systems retrieve memories before generation, post-retrieval follows a generate-then-retrieve pattern. The system first produces an intermediate representation, e.g., a topic, intent descriptor, hypothesized entities and relations, or a draft structure, and then retrieves based on that intermediate representation. For example, a model may generate a high-level topic prior to answering and use it to retrieve memories, or transform a query into an imagined subgraph and select candidate subgraphs that minimize graph distance [47]. This design is motivated by a common failure mode in interactive settings: user queries are often ambiguous or poorly specified, and even LLM-based query rewriting may not reliably map such queries onto the right memories. By interposing a topic-generation step, post-retrieval is less sensitive to superficial query phrasing [72].

Beyond explicit symbolic queries, there are also generative retrieval variants where the model uses its current latent reasoning state to generate a sequence of latent memory token representations. It then retrieves memories by embedding similarity in that latent space. The model can be trained using answer quality as the learning goal, encouraging latent memory token representations that surface more helpful memories [73].

For knowledge memory, the intermediate representation is often made closer to a canonical form so that the system can match rules and facts and then verify them. For experience memory, the intermediate representation often includes the missing context that the user did not say explicitly, such as the user goal, constraints, or what has already been tried. This is useful when the system must recover the right episode from many similar logs.

3) *Hybrid-source Retrieval*: Graph memory retrieval is increasingly studied in a hybrid-source setting, where the system coordinates internal memory with external resources. Here, external knowledge is treated as an additional retrievable resource alongside the memory store [69]. Examples include a local document index, online search APIs that return titles and snippets with URLs, and task environments that can be accessed through an agent interface. A key challenge is source selection. The system must decide when to rely on internal

memory versus external retrieval and how to merge evidence or resolve conflicts across sources.

Hybrid-source retrieval naturally supports knowledge memory and experience memory. External sources excel at providing information that changes frequently or extends beyond what the system has seen, which is closer to knowledge memory. Internal experience mainly provides personal and local details, such as users' previous purchase history or which tools failed in past attempts. When the two disagree, the merge rule should depend on what is being retrieved. If the system retrieves a fact, it should prioritize evidence that can be verified through multiple independent sources and traced back to authoritative origins, since outdated or unreliable external data is worse than trusted internal knowledge memory. If the system retrieves personal experience, it should prioritize internal records that match both the correct user and the correct time period, since external data cannot capture this individual [74].

VII. MEMORY EVOLUTION: LEARNING OVER TIME

As agent systems interact with dynamic environments over extended periods, their memory must not remain static but evolve to incorporate new information, resolve inconsistencies, and adapt to changing contexts. Graph-based memory structures are particularly well-suited for evolution due to their explicit modeling of relational connections, temporal dependencies, and validity that enable direct updates through node/edge/subgraph operations. Drawing inspiration from cognitive science, where human memory consolidates and adapts via mechanisms like synaptic plasticity, graph-based agent memory evolves through internal self-reflection and external exploration. This dual approach addresses conflicts between new and old memories (e.g., outdated facts versus recent observations) while enhancing long-term coherence and adaptability. As shown in Figure 8, we categorize memory evolution into two complementary paradigms: i) **internal self-evolving**, which focuses on intrinsic graph operations to maintain consistency without external input; ii) **external self-exploration**, which leverages active interaction with the environment to ground and refine memory automatically. These mechanisms transform passive knowledge repositories into proactive, learning systems.

A. Internal Self-Evolving

Internal self-evolving treats the agent's memory graph as a closed-loop system capable of introspective refinement. This process is analogous to human memory consolidation during sleep or rest, where the brain organizes recent experiences, abstracts general rules, and forgets trivial details without requiring new external sensory input. In the context of graph-based agent memory, this involves reorganizing the graph topology to enhance retrieval efficiency, logical consistency, and generalization capabilities. Unlike traditional storage that simply appends the logs, internal evolution applies structural transformations to the graph, typically focusing on three key aspects: **Memory Consolidation**, **Graph Reasoning** and **Graph Reorganization**.

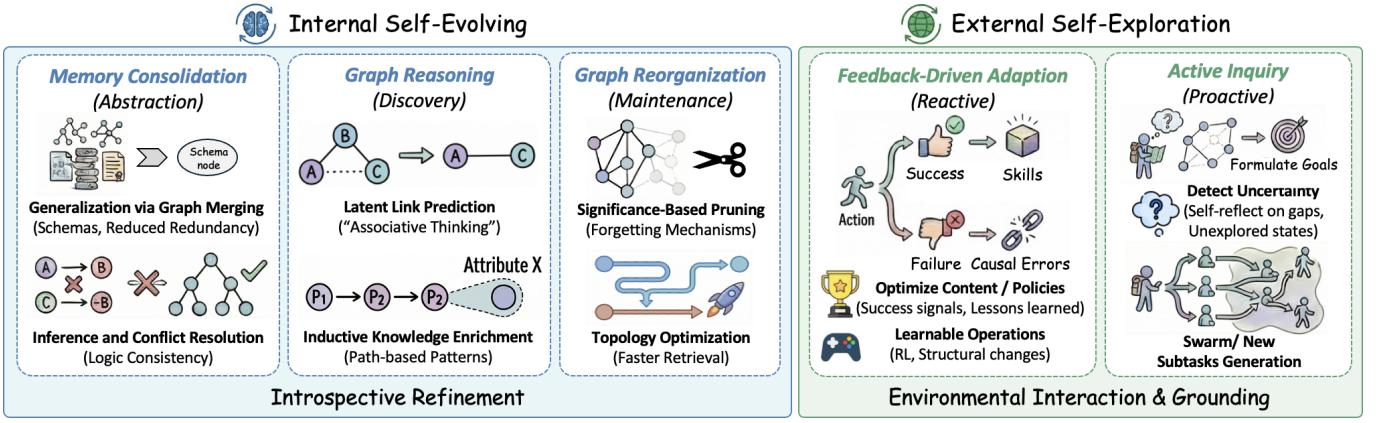


Fig. 8: A taxonomy of Memory Evolution mechanisms. (**Left**) **Internal Self-Evolving** entails introspective refinement via memory consolidation, graph reasoning, and reorganization. (**Right**) **External Self-Exploration** involves grounding memory through environmental interaction, categorized into reactive feedback-driven adaptation and proactive active inquiry.

1) **Memory Consolidation:** The core of internal evolution is the abstraction of high-level knowledge from raw experiential data. Agents accumulate vast amounts of experience memory (specific trajectories or interaction logs). Internal evolution mechanisms [14], [18], [25] analyze these episodes to induce generalized knowledge.

- **Generalization via Graph Merging:** When an agent observes multiple similar distinct events, it can merge these subgraph instances into a generalized schema node [75]. This reduces storage redundancy and creates a canonical representation of "skills" or "facts" [76], [77]. More advanced methods like Mem0 [31] and FLEX [78] employ LLM-based semantic gating, where the model evaluates the information gain of new trajectories

- **Inference and Conflict Resolution:** Through internal reasoning (e.g., logical inference or graph traversal), the agent can detect implicit contradictions between nodes. For instance, if node A implies B , but node C implies $\neg B$, the system triggers a self-correction process to resolve the conflict based on confidence scores or temporal recency, updating the graph structure to maintain logical coherence [24], [79], [80].

2) **Graph Reasoning:** Beyond consolidating existing nodes, internal evolution also entails discovering latent connections to address memory sparsity. Agents actively scan their memory graphs to identify missing edges or infer new relationships that were not explicitly observed but can be deduced from existing facts. This process transforms the memory from a sparse set of isolated trajectories into a densely connected knowledge web.

- **Latent Link Prediction:** Utilizing the semantic capabilities of LLMs, agents can predict potential relationships between disjoint subgraphs. For instance, if the memory contains $(A \xrightarrow{\text{cause}} B)$ and $(B \xrightarrow{\text{cause}} C)$, the agent can autonomously infer and insert a transitive edge $(A \xrightarrow{\text{leads_to}} C)$ [79], [82]. This mechanism mimics the "associative thinking" in human cognition, allowing agents to "connect the dots" between temporally distant events without new external input.

• **Inductive Knowledge Enrichment:** Agents can employ path-based reasoning to derive new attributes or facts. By traversing the graph structure, the system identifies patterns and enriches the graph with inferred "covariates" or high-order relationships [75]. This effectively mitigates the incompleteness of the initial memory construction, ensuring that future retrieval can access logically derivative information that was never explicitly written into the logs.

3) **Graph Reorganization:** Infinite accumulation of memory leads to retrieval latency and noise. Internal self-evolving implements "forgetting" mechanisms akin to biological systems to maintain the graph's health and quality.

- **Significance-Based Pruning:** Agents employ algorithms (e.g., PageRank variants or decay functions) to evaluate the utility of memory nodes. Nodes that are rarely accessed or have low contribution to recent decision-making are pruned or compressed [62], [81], [83].

- **Topology Optimization:** This involves restructuring the edges to shorten the path length between frequently associated concepts [76], [91]. By increasing the edge weights or creating shortcuts between correlated nodes, the agent optimizes the graph for faster future retrieval, compiling its experience into a more efficient structure.

B. External Self-Exploration

While internal evolution refines memory based on intrinsic consistency, it cannot verify the *validity* of knowledge against the real world. **External Self-Exploration** bridges this gap by grounding memory evolution in environmental interaction. Instead of passively recording all events, effective exploration uses environmental feedback (e.g., success signals, errors) to distinguish signal from noise and proactively seeks missing information. We categorize this broadly into feedback-driven adaptation (reactive) and active inquiry (proactive).

The first paradigm, **feedback-driven adaptation**, focuses on optimizing memory content and management policies based on the outcomes of past actions. In open-ended environments, raw interaction logs are often noisy; therefore, agents utilize

task execution results as supervisory signals to distill actionable knowledge. Methods like ExpeL [88] and Matrix [84] differentiate processing based on success or failure: successful trajectories are crystallized into reusable skills, while failed ones undergo comparative analysis to isolate causal errors, explicitly encoding “lessons learned” to favor high-utility behaviors. This evolutionary logic extends beyond content to the memory management policy itself. Rather than relying on fixed rules, systems like Memory-R1 [54] and Inside Out [85] treat memory operations (such as addition or deletion) as learnable actions within a Reinforcement Learning framework. By receiving rewards for downstream task accuracy, the agent autonomously learns an optimal policy for maintaining the graph. Similarly, MemEvolve [86] applies this feedback-driven principle to the architectural level, dynamically adjusting storage structures and indexing mechanisms to tailor the system to the changing complexity of the deployment environment.

However, relying solely on reactive adaptation suffers from coverage bias, where the agent’s knowledge remains limited to the specific tasks it has been assigned. To overcome this, the second paradigm, **active inquiry**, transitions the agent from a passive learner to a proactive explorer. Advanced frameworks empower agents to detect uncertainty in their current graph and autonomously formulate goals to resolve it. For instance, ProMem [89] enables agents to critically self-reflect on missing nodes or ambiguous edges, subsequently generating specific queries to fill these knowledge gaps. AgentEvolver [90] expands this capability to the task space by autonomously generating new sub-tasks that force navigation through unexplored states, preemptively populating memory with diverse experiences. To further accelerate this process, KIMI K2.5 [87] introduces a scalable swarm mechanism, spawning parallel sub-agents to explore disjoint branches of a problem space. This allows the central memory to rapidly ingest diverse perspectives and edge cases, transforming the system from a passive container of experience into an active constructor of knowledge.

Despite progress in feedback-driven adaptation and active inquiry, explicitly leveraging graph structure for exploration remains underexplored. Here, we give some suggestions about using the graph in self-exploration. First, topology-guided exploration can prioritize sparsely-connected clusters or bridge disconnected subgraphs to systematically improve knowledge coverage and connectivity. Then, multi-granularity strategies coordinate discovery across seeking high-level relational patterns while populating concrete instances through targeted interactions. These approaches transform memory graphs from passive repositories into active architectures that dynamically construct and refine their own knowledge boundaries.

VIII. OPEN-SOURCED LIBRARIES AND BENCHMARKS

A. Open-sourced Libraries

In Table IV in Appendix B, we provide a systematic comparison of eleven representative open-source memory libraries across key functional dimensions.

Several libraries offer graph-based memory representation and utilization, including Cognee [92], Mem0 [31], OpenMemory, MemMachine, Memary, and Graphiti. Graph-based

structures naturally couple with structured retrieval, enabling multi-hop or relational queries essential for reasoning over entities, events, or concepts. In terms of functional coverage, OpenMemory and Mem0 [31] act as the most comprehensive graph memory tools, supporting memory construction, interaction-driven updates, lifecycle management, temporal awareness, and graph management. Graph-based memory tools can be directly integrated into agent architectures to dynamically leverage structured knowledge, supporting long-term retrieval, temporal reasoning, and multi-step task execution. Cognee [92] provides queryable graph embeddings, Mem0 [31] and OpenMemory support session-aware memory updates, and Graphiti enables temporal graph reasoning for multi-step planning. For non-graph memory systems, memory construction is predominantly interaction or session-driven, such as LangMem, LightMem [93], and O-Mem [60], with mechanisms to update memory incrementally and support retrieval. While they provide retrieval functionalities fundamentally based on embedding similarity. Although lightweight tools like Memori and MemMachine focus on modular memory management, prioritizing ease of integration and supporting agent conditioning through APIs.

B. Datasets and Benchmarks

Unlike standard NLP tasks, evaluating *agent memory* should account for information dispersed throughout extended interactions and evolving system environments. Effective benchmarks in this area prioritize an agent’s ability to reuse observed data despite the limitations of finite context windows and computational costs. In the following, we highlight key benchmarks tailored for memory-augmented agents. These selections, summarized in Table I, are evaluated against a unified set of criteria including modality, environment realism, and memory type to provide a comparative benchmark overview.

a) Scenario taxonomy: We categorize existing benchmarks through a scenario-based taxonomy that reflects the diverse application settings of memory-augmented agents. This classification is based on three key aspects: interaction patterns ranging from multi-turn dialogues to long-horizon tasks, working interfaces including web-based or embodied and tool-assisted environments, and the time spans of information reuse. Following this structure, we identify seven representative scenarios: Interaction, Personalization, Web, LongContext, Continual, Environments, and Tool/Gen.

1) Interaction: Multi-turn and Cross-session Conversational Memory: Benchmarks in the Interaction scenario focus on an agent’s ability to maintain continuity across multi-turn and cross-session dialogues. In these settings, relevant information introduced early in the conversation must be accurately recalled and applied later. This category includes datasets such as LoCoMo [94], LongMemEval [95], MemoryAgentBench [96], MEMTRACK [97], MADial-Bench [98], MemSim [99], ChMapData [100], MSC [101], MMRC [102], MemBench [103], StoryBench [104], DialSim [105], and RealMem [106]. These benchmarks prioritize long-range context, consistency across different sessions, and the retrieval of previously mentioned user-provided facts during an ongoing dialogue.

TABLE I: Agent memory benchmarks grouped by scenario.

Name	Scenario	Modality	Feature	Environment	Memory type	Link(Feb 2026)
LoCoMo [94]	Interaction	Text+Image	Long conversational memory	real	Factual	• Website
LongMemEval [95]	Interaction	Text	Long-term interactive memory	simulated	Factual	• GitHub
MemoryAgentBench [96]	Interaction	Text	Multi-turn interactions	simulated	Factual + Experiential	• GitHub
MEMTRACK [97]	Interaction	Text+Code+Logs	Long-term interactive memory	simulated	Factual + Experiential	• Website
MADial-Bench [98]	Interaction	Text	Memory-augmented dialogue generation	simulated	Factual	• GitHub
MemSim [99]	Interaction	Text	Bayesian memory simulation	simulated	Factual + Experiential	• GitHub
ChMapData [100]	Interaction	Text	Memory-aware proactive dialogue	simulated	Factual	• GitHub
MSC [101]	Interaction	Text	Multi-session chat	simulated	Factual	• Website
MMRC [102]	Interaction	Text+Image	Multi-modal real-world conversation	simulated	Factual	• GitHub
MemBench [103]	Interaction	Text	Interactive scenarios	simulated	Factual + Experiential	• GitHub
StoryBench [104]	Interaction	Text	Interactive fiction memory	mixed	Factual + Experiential	• Website
DialSim [105]	Interaction	Text	Multi-dialogue understanding	real	Factual + Experiential	• Website
RealMem [106]	Interaction	Text	Project-oriented long-term memory interaction	simulated	Factual + Experiential	• GitHub
PersonaMem [107]	Personalization	Text	Dynamic user profiling	simulated	Factual	• GitHub
PerLTQA [108]	Personalization	Text	Social personalized interactions	simulated	Factual	• Website
MemoryBank [81]	Personalization	Text	User memory updating	simulated	Factual	• GitHub
MPR [109]	Personalization	Text	User personalization	simulated	Factual	• GitHub
PrefEval [110]	Personalization	Text	Personal preferences	simulated	Factual	• Website
LOCCO [111]	Personalization	Text	Chronological conversations	simulated	Factual	• GitHub
WebChoreArena [112]	Web	Text+Image	Tedious web browsing	real	Factual + Experiential	• GitHub
MT-Mind2Web [113]	Web	Text	Conversational web navigation	real	Factual + Experiential	• GitHub
WebShop [114]	Web	Text+Image	E-commerce web interaction	simulated	Experiential	• GitHub
WebArena [115]	Web	Text+Image	Web interaction	real	Experiential	• GitHub
MMInA [116]	Web	Text+Image	Multihop web agent	real	Factual + Experiential	• Website
NQ [117]	LongContext	Text	Natural question answering	simulated	Factual	• Website
TriviaQA [118]	LongContext	Text	Large-scale question answering	simulated	Factual	• Website
PopQA [119]	LongContext	Text	Adaptive retrieval augmentation	simulated	Factual	• GitHub
HotpotQA [120]	LongContext	Text	Explainable multi-hop QA	simulated	Factual	• Website
2wikimultihopQA [121]	LongContext	Text	Multi-hop QA	simulated	Factual	• GitHub
Musique [122]	LongContext	Text	Multi-hop QA	simulated	Factual	• GitHub
LongBench [123]	LongContext	Text	Long-context understanding	mixed	Factual	• GitHub
LongBench v2 [124]	LongContext	Text	Long-context multitasks	mixed	Factual	• GitHub
RULER [125]	LongContext	Text	Long-context retrieval	simulated	Factual	• GitHub
BABILong [126]	LongContext	Text	Long-context reasoning	simulated	Factual	• GitHub
MM-Needle [127]	LongContext	Text+Image	Multimodal needle retrieval	simulated	Factual	• Website
HaluMem [128]	LongContext	Text	Memory hallucination eval	simulated	Factual	• GitHub
MemoryBench [129]	Continual	Text	Continual learning	simulated	Factual + Experiential	• GitHub
LifelongAgentBench [130]	Continual	Text	Lifelong learning	simulated	Factual + Experiential	• Website
StreamBench [131]	Continual	Text	Continuous online learning	simulated	Factual + Experiential	• Website
Evo-Memory [132]	Continual	Text	Test-time learning	simulated	Factual + Experiential	• Website
Ego4D [133]	Environments	Video+Audio	Egocentric episodic memory	real	Experiential	• Website
EgoLife [134]	Environments	Video+Audio	Long-context life QA	real	Experiential	• Website
ALFWorld [135]	Environments	Text	Household tasks	simulated	Factual + Experiential	• Website
BabyAI [136]	Environments	Text	Language navigation	simulated	Experiential	• Website
ScienceWorld [137]	Environments	Text	Multi-step science experiments	simulated	Factual + Experiential	• GitHub
AgentGym [138]	Environments	Text	Multiple environments	mixed	Experiential	• Website
AgentBoard [139]	Environments	Text	Multi-round interaction	mixed	Experiential	• GitHub
SWE-Bench [140]	Tool/Gen	Text+Code	Code repair	real	Experiential	• Website
GAIA [141]	Tool/Gen	Text	Deep research tasks	real	Experiential	• Website
xBench-DS [142]	Tool/Gen	Text+Image	Deep-search evaluation	real	Experiential	• Website
ToolBench [143]	Tool/Gen	Text→API	API tool use	real	Experiential	• Website
GenAI-Bench [144]	Tool/Gen	Text+Image	Visual generation eval	real	Experiential	• Website

The evaluation typically centers on memory recall over extended histories, contextual reuse of information to produce coherent responses, and consistency maintenance to avoid self-contradiction. These capabilities are fundamental for assistant-style agents where memory failures directly lead to a degraded user experience. Common metrics include task-level accuracy, retrieval-oriented measures like Recall@k, and dialogue-level consistency rates. Furthermore, some benchmarks track success rates over multi-turn tasks to see if recalled information is effectively integrated into responses. A notable limitation is that many Interaction benchmarks lack explicit supervision for memory updates when dealing with conflicting facts. Consequently, the mechanism by which agents overwrite or forget outdated information is less systematically evaluated than their ability to recall it.

2) *Personalization: User Profiling, Preferences, and Memory Updates:* Personalization benchmarks examine an agent's

ability to manage persistent user-centric facts and profile attributes, as seen in PersonaMem [107], PerLTQA [108], MemoryBank [81], MPR [109], PrefEval [110], and LOCCO [111]. These benchmarks evaluate whether an agent can build a stable user model and integrate new user information over time. For real-world assistants, a primary challenge is avoiding persona drift where the agent forgets or contradicts previously established preferences. However, a current limitation of these tasks is their frequent reliance on clear supervision regarding what should be stored. In contrast, practical deployment necessitates selective writing and privacy-aware retention, both of which remain less addressed in existing efforts.

3) *Web: Long-horizon Browsing and Multi-step Online Tasks:* Web benchmarks evaluate memory within extended action trajectories where agents must track environmental states and intermediate results across numerous steps. Datasets like WebShop [114] and WebArena [115] focus on e-commerce and

functional website interactions. WebChoreArena [112] targets complex browsing routines, while MT-Mind2Web [113] emphasizes conversational navigation and MMInA [116] assesses multi-hop web interactions. These tasks place heavy demands on experiential memory because agents must often cache page states to prevent redundant actions. They also highlight the necessity of resource-efficient memory given that excessive tool calls can lead to high operational costs. A prevailing challenge is that success in these benchmarks can sometimes be achieved via simple heuristics, meaning that isolating the specific impact of memory requires controlled settings such as limiting memory capacity.

4) LongContext: Long-document Understanding and Retrieval: LongContext benchmarks measure agent performance under high-volume inputs and retrieval-intensive settings. Established QA suites including NQ [117], TriviaQA [118], PopQA [119], HotpotQA [120], 2wikimultihopQA [121], and Musique [122] test evidence aggregation and multi-step reasoning. More recent frameworks like LongBench [123] and LongBench v2 [124] offer multi-task evaluations, while RULER [125], BABILong [126], MM-Needle [127], and HaluMem [128] focus on needle-in-a-haystack retrieval and hallucination evaluation. While these benchmarks are essential for modeling evidence access, they are not always perfect indicators of agent memory. Many of these tasks remain single-turn and do not require the agent to actively write to a persistent memory store, potentially conflating long-context processing with dedicated memory mechanisms.

5) Continual: Lifelong Learning and Test-time Adaptation: Continual benchmarks assess whether agents can improve over time without experiencing catastrophic forgetting, typically under streaming or sequential task distributions. Frameworks such as MemoryBench [129], LifelongAgentBench [130], StreamBench [131], and Evo-Memory [132] capture elements of online updates and test-time adaptation. This category represents lifelong memory in its strictest sense and requires models to maintain proficiency in earlier tasks while acquiring new knowledge. Despite its importance, this area lacks standardized reporting because metrics for forgetting and transfer gains vary significantly. Furthermore, it is often difficult to discern whether performance gains stem from parametric updates or retrieval over past logs.

6) Environments: Embodied and Interactive Worlds: Environment-based benchmarks evaluate agents in simulated or physical interactive settings where memory must distill observations under partial observability. Ego4D [133] and EgoLife [134] focus on egocentric episodic memory and multimodal life-logging. ALFWORLD [135] and BabyAI [136] emphasize instruction following and navigation, while ScienceWorld [137] tests multi-step experimentation. Broader suites like AgentGym [138] and AgentBoard [139] offer multi-round evaluations using planning-centric analysis. These benchmarks primarily test experiential memory and robustness across environmental variations. However, since performance is often tied to environment-specific skills, claiming memory-related benefits requires carefully controlling for planning and tool-use variables.

7) Tool/Gen: Tool Use and Workflow Execution: Tool/Gen benchmarks evaluate memory within workflows involving external APIs and iterative reasoning. ToolBench [143] focuses on API invocation, while SWE-Bench [140] targets software engineering through iterative debugging. GAIA [141], xBench-DS [142], and GenAI-Bench [144] measure complex research and generation behaviors. These tasks emphasize process memory or the ability to retain intermediate hypotheses and failed attempts. They also underscore operational issues, e.g., traceability and the financial cost of retries. A significant hurdle is evaluation complexity because success hinges on environment stability and the design of specialized scoring scripts.

a) Synthesis of Evaluation Landscapes: Overall, these benchmarks provide a broad perspective on agent capabilities by testing them in interactive and long-horizon tasks. While memory is not always the only metric, success in these environments depends on the ability of an agent to store relevant data and use past experience for current decisions. Future studies can improve these evaluations by measuring the specific impact of memory through ablation tests and focusing on how agents handle changing information in dynamic settings. By using clear efficiency metrics and ensuring that environments are reproducible, these benchmarks help provide a better understanding of memory behaviors in practice.

b) Strategic Benchmark Selection: Choosing the right benchmark depends on which memory capability is being studied. Interaction and Personalization datasets are best for testing conversational persistence. Web and Environments are more suitable for evaluating how agents manage long sequences of actions and experiential data. LongContext tasks remain the standard for checking fact retrieval in large inputs. For research on agents that learn over time, Continual benchmarks show how well information is kept, while Tool/Gen tasks evaluate memory during the execution of complex technical steps.

IX. APPLICATIONS

The applicability of (graph-based) agent memory ranges from conversational chatbots to embodied robots and science agents. By addressing challenges, including long-term knowledge retention, personalized interaction, multi-step reasoning, and self-evolution, memory can enhance the effectiveness and reliability of LLM agents in a wide range of application domains. This section systematically discusses both current and prospective applications.

A. Conversational Agents

Conversational agents, such as Claude¹ and ChatGPT², are one of the most common use cases of LLM-based systems. The agents face challenges in maintaining coherent and personalized dialogues within a context for a long period of time in a multi-session dialogue and require sophisticated memory systems to effectively update their knowledge and user preferences.

Early studies focused on the controllability and stability of memory systems. The Memory Sandbox [145] and LD-Agent [146] initiatives laid the foundation by highlighting the

¹<https://claude.ai/>

²<https://chatgpt.com/>

transparency of memory systems and the distinction between event memory and personality to enhance the credibility of the memory system. Contemporary studies on memory moved on to tackle the issue of context fragmentation in multi-session dialogue systems by using structural optimization methods. SeCom [147] optimizes the dialogue structure by extracting topics in dialogue systems, while SGMem [35] relies on semantic graphs to connect fragmented dialogue sessions. These studies upgrade memory systems from simple linear structures to knowledge graphs enabling more precise memory recall.

Aside from memory storage, advanced memory-based dialogue agents need to be equipped with dynamic reasoning and time evolution capabilities. RMM [57] extends memory systems with reflection-based self-correction mechanisms, whereas TReMu [37] uses temporal knowledge graphs to capture intricate time-based relationships. Nevertheless, the trend of increasing memory system complexity is challenged by the recent ENGRAM [34] project, which showed that an efficient memory system with a basic typed memory structure consisting of episodic, semantic, and procedural memory types can achieve comparable performance to complex memory systems.

B. Code Agents

The software engineering processes for code generation [148] and code simulation [149] pose a unique challenge for the memory component of the agent, as they require following rigid structural requirements and logical flow of software programming. In the initial stages, the problem of task confusion was addressed in both MetaGPT [150] and ChatDev [151] by modeling conventional human workflows and definitions of roles and responsibilities. SWE-agent [1] further enhanced this approach with the inclusion of Agent-Computer Interfaces, which allow the agent to perform operations on source control systems. However, as the task becomes more intricate, linear approaches are no longer sufficient. TALM [152] introduces a new paradigm, moving away from conventional linear workflows and towards a dynamic tree-based architecture. By using divide and conquer approaches and taking advantage of the agent's long-term memory, TALM demonstrates the need for a hierarchical structure, which can handle the non-linear dependencies involved in code generation.

In addition to this orchestration of tasks, agents should navigate the complex information space of software, which resembles a knowledge graph of dependencies and logic. While Reflexion [153] introduced a form of basic self-correction through a verbal feedback loop, recent research has focused on structural context. In this regard, RepoAudit [154] attempts to solve issues related to the repository. An auditing agent is introduced that explores the codebase on its own. This is akin to a graph traversal algorithm on file dependencies. As a result, an accurate analysis is guaranteed. MemGovern [155] and Multi-Agent RL Debugging [156] are extensions that arrange the external information gathered from the GitHub website and the feedback loop in the form of a retrievable knowledge base. This is a clear indication of the evolution of memory associated with software agents, which is moving away from simple memory to more structured memory that allows for basic reasoning about the topology.

C. Recommender Systems

Agent memory is applied in recommender systems to address the issue of long and dynamic user histories that are difficult to handle using recommendation agents [157]. In practice, recommendation agents often truncate histories, letting short-term noise, e.g., accidental interactions, override stable long-term preferences. Meanwhile, fine-tuning agent parameters to track drifting user preferences is costly [158]. External memory is a more efficient solution that alleviates the problem through the reduction of retraining and token costs.

The current agent-based recommender systems have adopted a three-level memory maintenance approach that starts with coarse-grained retrieval and ends with fine-grained reasoning. In the first level of history as memory, both MAP [159] and AgentCF++ [160] focus on scalability. For the latter, a dual-layer approach is proposed for noise filtering and social contextualization. For the second level of structured key-value memory, interactions are structured into semantic structures. MemoCRS [161] and Agent4Rec [162] use entity-key pairs for rating association. Finally, in the third level of memory as cognition, memory is made active by reflection and planning. At this level, CRAVE [163] and AgentCF [164] summarize preference rules. RecMind [165] applies strategic multi-step planning. In order to make these dynamic systems concrete and grounded in reality, KGLA [166] applies a Knowledge Graph for concrete metadata injection. MR.Rec [167] uses RL for adaptive memory retrieval and logical reasoning.

D. Financial Agents

Financial markets are considered to be challenging for agent memory systems due to the information decay pattern, heterogeneous data streams from various sources, and the need to balance the patterns with the latest market conditions [168]. Financial agents need memory systems with high priority for recency and the ability to preserve patterns. Moreover, financial decision-making also requires interpretability and risk management, where memory systems are essential for counterfactual reasoning, optimization, and adaptation.

The first attempts have been made in individual cognitive simulation, in which FinMem [44] tackles cognitive bottlenecks through a layered memory structure inspired by human traders' behavior. To address the natural bias in individual AI agents, TradingGPT [169] has proposed an extension to collaborative cognitive simulation, using inter-agent debate as a mechanism for de-biasing through collective intelligence. Progressing toward professional capabilities, FinCon [170] has set up a managerial structure for analysts, with agents empowered with higher memory capabilities to maintain a history of actions, profit-loss sequences, and changing investment beliefs for risk management strategies. Finally, FinAgent [171] bridges the information gap by extending memory to multimodal perception, enabling agents to process K-line charts and tool-augmented data for holistic market analysis.

Current implementations of memory-augmented financial agents remain relatively limited. Future graph-based memory holds transformative potential in finance, e.g., hierarchical graphs can enhance multi-asset portfolio management by

modeling cross-asset correlations, and temporal graphs can improve risk management and tail risk analysis.

E. Game Agents

The game environment poses a major challenge for agents' memory systems due to its dynamics, complex rules, and long-term goals—especially in the open world that requires multi-step reasoning and exploratory learning. Memory must store experiential and world knowledge (including successes and failures) and allow efficient real-time access to support skill acquisition. Early research on game agents focused on mastering open-ended environments like Minecraft through progressively richer memory and perception mechanisms. Ghost in the Minecraft (GITM) [70] employed dictionary-based memory to capture spatial layouts and crafting knowledge via text interactions. Voyager [4] advanced this paradigm by introducing lifelong learning, using an ever-growing library of executable code as procedural memory for skill acquisition and reuse. Jarvis-1 [172] further incorporated multimodal memory to align textual reasoning with visual perception, enabling mastery of over 200 tasks in Minecraft. Most recently, Optimus-1 [28] addressed long-horizon planning by organizing experience into a hierarchical directed knowledge graph coupled with an abstracted experience pool.

Recently, the paradigm has shifted towards generalist agents capable of operating across diverse environments via unified interfaces. Cradle [61] broke the boundaries of game-specific APIs by establishing a unified human-like interface by grounding interaction in screenshots and keyboard-mouse actions. This enables agents to play multiple commercial games and requires episodic memory to retain cross-interaction context and accumulated gameplay experience. Following this direction, SIMA [173] and SIMA 2 [174] focus on instructable agents capable of executing arbitrary natural-language commands across many 3D environments. The latest version further extends this paradigm with higher-level reasoning and self-improvement, moving from passive instruction following to active skill acquisition. These advances place increasing demands on memory systems, including long-term episodic memory and mechanisms for accumulating and reusing strategies across environments. Future research should focus on expressive memory architectures such as dynamic and temporal graph construction for latent skill hierarchy discovery and time-aware causal reasoning.

F. Robotics and Embodied Agents

Embodied agents must continuously ground their decision-making in the physical world, which is dynamic and partially observable, and perform long-horizon manipulation tasks that span multiple interaction episodes. Therefore, agents embodied in physical or virtual environments have special challenges that require advanced memory mechanisms. HELPER [175] and MAP-VLA [176] both leverage memory to bridge high-level language instructions and executable actions, but they have different ways, such as key-value memory to map natural language commands directly to robot code and a reusable memory library to retrieve and adapt specific manipulation strategies.

In a further step, STRAP [177] improves trajectory retrieval by introducing a flexible memory that uses dynamic time warping to match variable-length motion sub-sequences within large and diverse datasets. In contrast, TrackVLA++ [178] uses a dual memory architecture to target perception stability and address the long-term maintenance.

In conclusion, the memory systems used by embodied agents remain flat or weakly structured, making it difficult to perform complex relational reasoning and hierarchical abstraction. Future work could focus on graph-based memory architectures that represent spatial relations between objects, hierarchical task structures, and action–effect causality.

G. Medical and Health Agents

Healthcare agents need advanced memory systems to facilitate high-risk medical decision-making, longitudinal patient care, and evidence-based diagnosis. Memory helps the healthcare agents retain patient histories over multiple visits, integrate new knowledge in the medical field, perform differential diagnosis, and provide personalized patient care while ensuring safety and interpretability. Specifically, AgentClinic [179] and AgentMental [180] focus on simulating physician–patient interactions with a memory module to record and track diagnostic information over time, while AgentMental [180] employs a dynamic tree-structured memory to organize and manage medical knowledge and conversation data. The memory in healthcare agents reflects the importance of longitudinal medical histories and diagnostic accuracy in real clinical settings, facilitating agents to keep previous information to give comprehensive suggestions. In contrast, AgentHospital [181] emphasizes a full-process virtual hospital environment, where agents evolve through large-scale interactions across both successful and failed cases, supported by richer data integration and more comprehensive system functionalities.

Collectively, these studies underscore the importance of structured and interaction-aware memory for healthcare agents. Graph-based memory grounded in medical ontologies such as UMLS [182] enables multi-hop clinical reasoning, explicit modeling of drug interactions, and temporal tracking of disease progression and treatments. Such structured representations provide a promising foundation for more robust medical agents with improved contextual reasoning, diagnostic planning, and long-term decision support [183].

H. Science Agents

In scientific discovery, agents are increasingly shifting from passive data analysis to active experimental partners that can efficiently search vast spaces and call some specific tools. Agent memory enables the iterative integration of theory and experiment within complex scientific workflows, functioning as a dynamic workspace rather than static information retrieval. ChatNT [184], ChemCrow [185], and El Agente [186] represent domain-specific scientific research support agents. All focus on enhancing LLM capabilities for professional research tasks while they have different scopes for research works such as biological sequence reasoning, chemical analysis, and quantum chemistry simulations. These systems primarily function as

intelligent tools that assist researchers in localized stages of the scientific process rather than modeling the entire research lifecycle. Furthermore, some complex agents are established as system-level scientific agents with complex environments. In biological research domain, Biomni [187] advances beyond task-level assistance by supporting complex biomedical research workflows, enabling agents to automatically identify, compose, and execute multi-step experimental pipelines. Similaly, CREST [188] targets materials science field and closes the loop between computational reasoning and physical experimentation, enabling agents to iteratively generate hypotheses and validate them through robotic synthesis. Generally, VirtualLab [189] operates at the organizational level, simulating human research institutions through teams of collaborative agents, and represents a shift from individual scientific tools to collective scientific intelligence, as a comprehensive agent.

Overall, this progression of existing scientific agents reflects a transition from localized scientific assistance toward holistic and autonomous scientific systems and requires effective memory modules. As agent capabilities scale, memory requirements become correspondingly more complex, demanding structured integration, organization, and management of heterogeneous domain knowledge, for which graph-based memory structures provide a clear and effective solution.

X. LIMITATIONS AND FUTURE DIRECTIONS

Despite significant progress in graph-based agent memory, several fundamental challenges present critical opportunities for advancing the field.

The Quality of Memory Graph. The quality of the memory graph fundamentally constrains the performance, reliability, and adaptability of graph-based agent memory systems [190], [191]. Unlike traditional memory systems, where quality is often related primarily to factual accuracy in downstream tasks, graph-based memory should introduce multidimensional quality criteria, including structural, semantic, temporal, and operational aspects, each of which directly shapes agent capabilities [18], [103]. And there is a scarcity of metrics designed to explicitly evaluate the intrinsic quality of the memory graph [103], [192].

Scalability and Efficiency. As agents accumulate experiences over extended interactions, memory systems face computational bottlenecks, with graph operations exhibiting quadratic or worse complexity [15]. Future research should explore memory compression techniques specifically designed for graph structures [193], incremental update algorithms that avoid full recomputation [194], and approximate retrieval methods that trade precision for substantial efficiency gains [195]. Hardware acceleration through specialized graph processing units [196] and distributed architectures [197] enable the management of millions of nodes while maintaining rapid access.

Privacy Protection and Security. Personal assistant applications require robust protection of sensitive information while enabling meaningful personalization [9]. Graph-based memory structures introduce unique vulnerabilities where relational patterns may inadvertently expose private data through inference attacks. Critical research directions include: (1) developing

differential privacy mechanisms [198] tailored for graph memory systems; (2) federated architectures enabling on-device processing to minimize data exposure; and (3) secure multi-party computation protocols that allow agents to benefit from collective experiences without compromising individual privacy. Beyond privacy leakage, memory systems face emerging threats from adversarial attacks. Similar to prompt injection and data poisoning attacks against LLMs [199], [200], adversaries can manipulate memory contents to corrupt agent behavior or inject malicious knowledge. Defense mechanisms such as memory content validation, anomaly detection, and robust auditing protocols are essential to ensure memory integrity.

Dynamic Schema Learning and Knowledge Transfer.

Current graph schemas are often domain-specific with limited reusability, requiring substantial re-engineering for new applications [11]. Future systems should pursue dynamic schema learning, where agents automatically identify relevant entity types and relationship patterns from raw experiences. Meta-learning approaches could enable rapid adaptation to new domains [201], combining with universal graph ontologies [202] and domain-agnostic abstraction mechanisms, to facilitate effective knowledge transfer across tasks.

Interpretability and Trustworthy. For agents to be deployed in high-stakes domains, their memory systems must be both human-understandable and transparent in operation. Graph-based memory architectures offer unique advantages for interpretability: their explicit relational structures naturally align with human mental models, enabling users to inspect and comprehend how agents organize and utilize information [79], [82], [191]. Critical directions include developing memory provenance tracking systems, creating interactive visualization interfaces that allow users to explore memory graphs at multiple levels [145]. By ensuring human oversight and understanding of agent memory processes, the systems can foster appropriate trust calibration, enabling users to identify potential biases, verify critical information, and maintain control over agent behavior [180], [203].

Theoretical Foundations. Establishing rigorous mathematical frameworks remains essential for advancing the field. Priority areas include formal models that provide completeness and consistency guarantees, complexity analysis establishing theoretical bounds on construction and retrieval operations, and scaling laws of the memory-augmented AI agents [204]. Comparative analysis with human cognitive architectures could identify fundamental gaps and opportunities for architectural improvements aligned with biological memory systems [205].

Memory Coordination in Multi-Agent Systems. In multi-agent or agent-swarm settings, memory is no longer an isolated component but a shared resource that directly affects task completion and coordination efficiency. Ineffective memory sharing or inconsistent memory updates can lead to conflicting decisions. Designing mechanisms for memory synchronization, role-aware memory access, and scalable coordination remains an open challenge, especially under communication constraints.

XI. CONCLUSION

As LLM-based agents evolve toward increasingly autonomous and general-purpose systems, memory emerges

as a critical component. Graph-based memory architectures represent a paradigm shift from simple storage mechanisms to structured, relational representations that enable sophisticated reasoning, personalization, and continual learning. This survey comprehensively reviews agent memory from a graph-based perspective. First, it introduces an agent memory taxonomy, including short-term vs. long-term memory, knowledge vs. experience memory, non-structural vs. structural memory, with a focus on graph-based memory implementation. Second, it systematically analyzes key graph-based agent memory techniques by lifecycle, including extraction, storage, retrieval, and evolution. Third, it summarizes open-source libraries, datasets and benchmarks, and diverse application scenarios for self-evolving agent memory. Finally, it identifies challenges and future research directions. We hope this survey serves as a valuable resource for researchers advancing the frontiers of agent memory systems and for practitioners seeking to build more capable, reliable, and trustworthy AI agents.

REFERENCES

- [1] J. Yang, C. E. Jimenez, A. Wettig, K. Lieret, S. Yao, K. Narasimhan, and O. Press, “SWE-agent: agent-computer interfaces enable automated software engineering,” in *NeurIPS*, 2024.
- [2] Y. Lin, S. Tang, B. Lyu, J. Wu, H. Lin, K. Yang, J. Li, M. Xia, D. Chen, S. Arora *et al.*, “Goedel-prover: A frontier model for open-source automated theorem proving,” *arXiv preprint arXiv:2502.07640*, 2025.
- [3] Meta, A. Bakhtin, N. Brown, E. Dinan, G. Farina, C. Flaherty, D. Fried, A. Goff, J. Gray, H. Hu *et al.*, “Human-level play in the game of diplomacy by combining language models with strategic reasoning,” *Science*, 2022.
- [4] G. Wang, Y. Xie, Y. Jiang, A. Mandlekar, C. Xiao, Y. Zhu, L. Fan, and A. Anandkumar, “Voyager: An open-ended embodied agent with large language models,” *Transactions on Machine Learning Research*, 2024.
- [5] Y. Su, D. Yang, S. Yao, and T. Yu, “Language agents: Foundations, prospects, and risks,” in *EMNLP: Tutorial Abstracts*, 2024.
- [6] Z. Wang, Z. Cheng, H. Zhu, D. Fried, and G. Neubig, “What are tools anyway? a survey from the language model perspective,” in *COLM*, 2024.
- [7] Y. Qin, S. Liang, Y. Ye, K. Zhu, L. Yan, Y. Lu, Y. Lin, X. Cong, X. Tang, B. Qian, S. Zhao, L. Hong, R. Tian, R. Xie, J. Zhou, M. Gerstein, dahai li, Z. Liu, and M. Sun, “ToolLLM: Facilitating large language models to master 16000+ real-world APIs,” in *ICLR*, 2024.
- [8] T. Sumers, S. Yao, K. R. Narasimhan, and T. L. Griffiths, “Cognitive architectures for language agents,” *Transactions on Machine Learning Research*, 2024.
- [9] Y. Li, H. Wen, W. Wang, X. Li, Y. Yuan, G. Liu, J. Liu, W. Xu, X. Wang, Y. Sun *et al.*, “Personal LLM agents: Insights and survey about the capability, efficiency and security,” *arXiv preprint arXiv:2401.05459*, 2024.
- [10] C. Yang, X. Wang, Q. Zhang, Q. Jiang, and X. Huang, “Efficient integration of external knowledge to ILM-based world models via retrieval-augmented generation and reinforcement learning,” in *Findings, EMNLP*, 2025, pp. 9484–9501.
- [11] Y. Shang, Y. Li, K. Zhao, L. Ma, J. Liu, F. Xu, and Y. Li, “Agentsquare: Automatic LLM agent search in modular design space,” in *ICLR*, 2025.
- [12] H. Ye, T. Liu, A. Zhang, W. Hua, and W. Jia, “Cognitive mirage: A review of hallucinations in large language models,” *arXiv preprint arXiv:2309.06794*, 2023.
- [13] H. Yu, T. Chen, J. Feng, J. Chen, W. Dai, Q. Yu, Y.-Q. Zhang, W.-Y. Ma, J. Liu, M. Wang *et al.*, “Memagent: Reshaping long-context ILM with multi-conv rl-based memory agent,” *arXiv preprint arXiv:2507.02259*, 2025.
- [14] J. Nan, W. Ma, W. Wu, and Y. Chen, “Nemori: Self-organizing agent memory inspired by cognitive science,” *arXiv preprint arXiv:2508.03341*, 2025.
- [15] R. Zeng, J. Fang, S. Liu, and Z. Meng, “On the structural memory of ILM agents,” *arXiv preprint arXiv:2412.15266*, 2024.
- [16] Z. Jia, J. Li, Y. Kang, Y. Wang, T. Wu, Q. Wang, X. Wang, S. Zhang, J. Shen, Q. Li, S. Qi, Y. Liang, D. He, Z. Zheng, and S.-C. Zhu, “The AI hippocampus: How far are we from human memory?” *Transactions on Machine Learning Research*, 2025.
- [17] H. Sun and S. Zeng, “Hierarchical memory for high-efficiency long-term reasoning in ILM agents,” *arXiv preprint arXiv:2507.22925*, 2025.
- [18] P. Rasmussen, P. Palychuk, T. Beauvais, J. Ryan, and D. Chalef, “Zep: a temporal knowledge graph architecture for agent memory,” *arXiv preprint arXiv:2501.13956*, 2025.
- [19] R. B. Yousuf, A. Khatri, S. Xu, M. Sharma, and N. Ramakrishnan, “Can an ILM induce a graph? investigating memory drift and context length,” *arXiv preprint arXiv:2510.03611*, 2025.
- [20] R. Zeng, J. Fang, S. Liu, and Z. Meng, “On the structural memory of ILM agents,” *arXiv preprint arXiv:2412.15266*, 2024.
- [21] S. Liang, Y. Zhang, and Y. Guo, “Personaagent with graphrag: Community-aware knowledge graphs for personalized ILM,” *arXiv preprint arXiv:2511.17467*, 2025.
- [22] C. Huan, Z. Meng, Y. Liu, Z. Yang, Y. Zhu, Y. Yun, S. Li, R. Gu, X. Wu, H. Zhang *et al.*, “Scaling graph chain-of-thought reasoning: A multi-agent framework with efficient ILM serving,” *arXiv preprint arXiv:2511.01633*, 2025.
- [23] M. Hu, T. Chen, Q. Chen, Y. Mu, W. Shao, and P. Luo, “Hiagent: Hierarchical working memory management for solving long-horizon agent tasks with large language model,” in *Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, 2025.
- [24] N. Shinn, F. Cassano, A. Gopinath, K. Narasimhan, and S. Yao, “Reflexion: language agents with verbal reinforcement learning,” in *Proceedings of the 37th International Conference on Neural Information Processing Systems*, 2023.
- [25] Y. Wang, R. Takanobu, Z. Liang, Y. Mao, Y. Hu, J. McAuley, and X. Wu, “Mem-{}{alpha}: Learning memory construction via reinforcement learning,” *arXiv preprint arXiv:2509.25911*, 2025.
- [26] H. Shi, B. Xie, Y. Liu, L. Sun, F. Liu, T. Wang, E. Zhou, H. Fan, X. Zhang, and G. Huang, “Memoryvla: Perceptual-cognitive memory in vision-language-action models for robotic manipulation,” *arXiv preprint arXiv:2508.19236*, 2025.
- [27] J. H. Yeo, M. Kim, and Y. M. Ro, “Multi-temporal lip-audio memory for visual speech recognition,” *ICASSP 2023 - 2023 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, 2023.
- [28] Z. Li, Y. Xie, R. Shao, G. Chen, D. Jiang, and L. Nie, “Optimus-1: Hybrid multimodal memory empowered agents excel in long-horizon tasks,” *Advances in neural information processing systems*, 2024.
- [29] A. Modarressi, A. Köksal, A. Imani, M. Fayyaz, and H. Schütze, “Memilm: Finetuning ILMs to use an explicit read-write memory,” *arXiv preprint arXiv:2404.11672*, 2024.
- [30] P. Anokhin, N. Semenov, A. Sorokin, D. Evseev, A. Kravchenko, M. Burtsev, and E. Burnaev, “Arigraph: Learning knowledge graph world models with episodic memory for ILM agents,” *arXiv preprint arXiv:2407.04363*, 2024.
- [31] P. Chhikara, D. Khant, S. Aryan, T. Singh, and D. Yadav, “Mem0: Building production-ready ai agents with scalable long-term memory,” *arXiv preprint arXiv:2504.19413*, 2025.
- [32] H. Yang, J. Chen, M. Siew, T. Lorido-Botran, and C. Joe-Wong, “ILM-powered decentralized generative agents with adaptive hierarchical knowledge graph for cooperative planning,” *arXiv preprint arXiv:2502.05453*, 2025.
- [33] G. Zhang, M. Fu, G. Wan, M. Yu, K. Wang, and S. Yan, “G-memory: Tracing hierarchical memory for multi-agent systems,” *arXiv preprint arXiv:2506.07398*, 2025.
- [34] D. Patel and S. Patel, “Engram: Effective, lightweight memory orchestration for conversational agents,” *arXiv preprint arXiv:2511.12960*, 2025.
- [35] Y. Wu, Y. Zhang, S. Liang, and Y. Liu, “Sgmem: Sentence graph memory for long-term conversational agents,” *arXiv preprint arXiv:2509.21212*, 2025.
- [36] Z. Wang, Z. Li, Z. Jiang, D. Tu, and W. Shi, “Crafting personalized agents through retrieval-augmented generation on editable memory graphs,” in *Proceedings of the 2024 Conference on Empirical Methods in Natural Language Processing*, 2024.
- [37] Y. Ge, S. Romeo, J. Cai, R. Shu, Y. Benajiba, M. Sunkara, and Y. Zhang, “Tremu: Towards neuro-symbolic temporal reasoning for ILM-agents with memory in multi-session dialogues,” in *Findings of the Association for Computational Linguistics: ACL 2025*, 2025.

- [38] X. Tan, X. Wang, X. Xu, X. Yuan, L. Zhu, and W. Zhang, “Memotime: Memory-augmented temporal knowledge graph enhanced large language model reasoning,” *arXiv preprint arXiv:2510.13614*, 2025.
- [39] H. Luo, G. Chen, Y. Zheng, X. Wu, Y. Guo, Q. Lin, Y. Feng, Z. Kuang, M. Song, Y. Zhu *et al.*, “Hypergraphrag: Retrieval-augmented generation via hypergraph-structured knowledge representation,” *arXiv preprint arXiv:2503.21322*, 2025.
- [40] S. Huang, H. Li, Y. Gu, X. Hu, Q. Li, and G. Xu, “Hyperg: Hypergraph-enhanced llms for structured knowledge,” in *Proceedings of the 48th International ACM SIGIR Conference on Research and Development in Information Retrieval*, 2025.
- [41] J. Jiang, K. Zhou, W. X. Zhao, Y. Song, C. Zhu, H. Zhu, and J.-R. Wen, “Kg-agent: An efficient autonomous agent framework for complex reasoning over knowledge graph,” in *Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, 2025.
- [42] L. Long, Y. He, W. Ye, Y. Pan, Y. Lin, H. Li, J. Zhao, and W. Li, “Seeing, listening, remembering, and reasoning: A multimodal agent with long-term memory,” *arXiv preprint arXiv:2508.09736*, 2025.
- [43] R. Salama, J. Cai, M. Yuan, A. Currey, M. Sunkara, Y. Zhang, and Y. Benajiba, “Memindsight: Autonomous memory augmentation for llm agents,” *arXiv preprint arXiv:2503.21760*, 2025.
- [44] Y. Yu, H. Li, Z. Chen, Y. Jiang, Y. Li, J. W. Suchow, D. Zhang, and K. Khashanah, “Finmem: A performance-enhanced llm trading agent with layered memory and character design,” *IEEE Transactions on Big Data*, 2025.
- [45] Y. Hou, H. Tamoto, and H. Miyashita, “‘my agent understands me better’: Integrating dynamic human-like memory recall and consolidation in llm-based agents,” in *Extended Abstracts of the CHI Conference on Human Factors in Computing Systems*, 2024.
- [46] M. Fisher, “Neural graph memory: A structured approach to long-term memory in multimodal agents,” 2025.
- [47] Y. Cai, Z. Guo, Y. Pei, W. Bian, and W. Zheng, “Simrag: Leveraging similar subgraphs for knowledge graphs driven retrieval-augmented generation,” in *Findings of the Association for Computational Linguistics: ACL 2025*, 2025.
- [48] X. Wu, Y. Shen, C. Shan, K. Song, S. Wang, B. Zhang, J. Feng, H. Cheng, W. Chen, Y. Xiong *et al.*, “Can graph learning improve planning in llm-based agents?” *Advances in Neural Information Processing Systems*, 2024.
- [49] H. Sun and S. Zeng, “Hierarchical memory for high-efficiency long-term reasoning in llm agents,” *arXiv preprint arXiv:2507.22925*, 2025.
- [50] A. Jonelagadda, C. Hahn, H. Zheng, and S. Penachio, “Mnemosyne: An unsupervised, human-inspired long-term memory architecture for edge-based llms,” *arXiv preprint arXiv:2510.08601*, 2025.
- [51] K. Zhang, X. Zhang, E. Ahmed, H. Jiang, C. Kumar, K. Sun, Z. Lin, S. Sharma, S. Oraby, A. Colak *et al.*, “Assomem: Scalable memory qa with multi-signal associative retrieval,” *arXiv preprint arXiv:2510.10397*, 2025.
- [52] Z. Huang, Z. Tian, Q. Guo, F. Zhang, Y. Zhou, D. Jiang, and X. Zhou, “Licomemory: Lightweight and cognitive agentic memory for efficient long-term reasoning,” *arXiv preprint arXiv:2511.01448*, 2025.
- [53] H. Zhou, Y. Chen, S. Guo, X. Yan, K. H. Lee, Z. Wang, K. Y. Lee, G. Zhang, K. Shao, L. Yang *et al.*, “Memento: Fine-tuning llm agents without fine-tuning llms,” *arXiv preprint arXiv:2508.16153*, 2025.
- [54] S. Yan, X. Yang, Z. Huang, E. Nie, Z. Ding, Z. Li, X. Ma, K. Kersting, J. Z. Pan, H. Schütze *et al.*, “Memory-rl: Enhancing large language model agents to manage and utilize memories via reinforcement learning,” *arXiv preprint arXiv:2508.19828*, 2025.
- [55] S. Xia, Z. Xu, J. Chai, W. Fan, Y. Song, X. Wang, G. Yin, W. Lin, H. Zhang, and J. Wang, “From experience to strategy: Empowering llm agents with trainable graph memory,” *arXiv preprint arXiv:2511.07800*, 2025.
- [56] M. Xu, G. Liang, K. Chen, W. Wang, X. Zhou, M. Yang, T. Zhao, and M. Zhang, “Memory-augmented query reconstruction for llm-based knowledge graph reasoning,” *arXiv preprint arXiv:2503.05193*, 2025.
- [57] Z. Tan, J. Yan, I.-H. Hsu, R. Han, Z. Wang, L. Le, Y. Song, Y. Chen, H. Palangi, G. Lee *et al.*, “In prospect and retrospect: Reflective memory management for long-term personalized dialogue agents,” in *Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, 2025.
- [58] T. Kim, V. François-Lavet, and M. Cochez, “Leveraging knowledge graph-based human-like memory systems to solve partially observable markov decision processes,” *arXiv preprint arXiv:2408.05861*, 2024.
- [59] R. Wang, S. Liang, Q. Chen, Y. Huang, M. Li, Y. Ma, D. Zhang, K. Qin, and M.-F. Leung, “Graphcogent: Mitigating llms’ working memory constraints via multi-agent collaboration in complex graph understanding,” *arXiv preprint arXiv:2508.12379*, 2025.
- [60] P. Wang, M. Tian, J. Li, Y. Liang, Y. Wang, Q. Chen, T. Wang, Z. Lu, J. Ma, Y. E. Jiang *et al.*, “Omni memory system for personalized, long horizon, self-evolving agents,” *arXiv preprint arXiv:2511.13593*, 2025.
- [61] W. Tan, W. Zhang, X. Xu, H. Xia, Z. Ding, B. Li, B. Zhou, J. Yue, J. Jiang, Y. Li, R. An, M. Qin, C. Zong, L. Zheng, Y. Wu, X. Chai, Y. Bi, T. Xie, P. Gu, X. Li, C. Zhang, L. Tian, C. Wang, X. Wang, B. F. Karlsson, B. An, S. YAN, and Z. Lu, “Cradle: Empowering foundation agents towards general computer control,” in *ICML*, 2025.
- [62] C. Packer, S. Wooders, K. Lin, V. Fang, S. G. Patil, I. Stoica, and J. E. Gonzalez, “Memgpt: Towards llms as operating systems,” *arXiv preprint arXiv:2310.08560*, 2023.
- [63] M. Hu, T. Chen, Q. Chen, Y. Mu, W. Shao, and P. Luo, “Hiagent: Hierarchical working memory management for solving long-horizon agent tasks with large language model,” in *Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, 2025.
- [64] Z. Jia, J. Li, X. Qu, and J. Wang, “Enhancing multi-agent systems via reinforcement learning with llm-based planner and graph-based policy,” *arXiv preprint arXiv:2503.10049*, 2025.
- [65] A. Rezazadeh, Z. Li, A. Lou, Y. Zhao, W. Wei, and Y. Bao, “Collaborative memory: Multi-user memory sharing in llm agents with dynamic access control,” *arXiv preprint arXiv:2505.18279*, 2025.
- [66] B. Yan, C. Li, H. Qian, S. Lu, and Z. Liu, “General agentic memory via deep research,” *arXiv preprint arXiv:2511.18423*, 2025.
- [67] Y. Zhang, W. Yuan, and Z. Jiang, “Bridging intuitive associations and deliberate recall: Empowering llm personal assistant with graph-structured long-term memory,” in *Findings of the Association for Computational Linguistics: ACL 2025*, 2025.
- [68] J. Liu, Z. Kong, C. Yang, F. Yang, T. Li, P. Dong, J. Nanjekye, H. Tang, G. Yuan, W. Niu *et al.*, “Rcr-router: Efficient role-aware context routing for multi-agent llm systems with structured memory,” *arXiv preprint arXiv:2508.04903*, 2025.
- [69] Q. Yuan, J. Lou, Z. Li, J. Chen, Y. Lu, H. Lin, L. Sun, D. Zhang, and X. Han, “Memsearcher: Training llms to reason, search and manage memory via end-to-end reinforcement learning,” *arXiv preprint arXiv:2511.02805*, 2025.
- [70] X. Zhu, Y. Chen, H. Tian, C. Tao, W. Su, C. Yang, G. Huang, B. Li, L. Lu, X. Wang *et al.*, “Ghost in the minecraft: Generally capable agents for open-world environments via large language models with text-based knowledge and memory,” *arXiv preprint arXiv:2305.17144*, 2023.
- [71] D. Han, C. Couturier, D. M. Diaz, X. Zhang, V. Röhle, and S. Rajmohan, “Legomem: Modular procedural memory for multi-agent llm systems for workflow automation,” *arXiv preprint arXiv:2510.04851*, 2025.
- [72] Y. Wang and X. Chen, “Mirix: Multi-agent memory system for llm-based agents,” *arXiv preprint arXiv:2507.07957*, 2025.
- [73] G. Zhang, M. Fu, and S. Yan, “Memgen: Weaving generative latent memory for self-evolving agents,” *arXiv preprint arXiv:2509.24704*, 2025.
- [74] Z. Zhou, A. Qu, Z. Wu, S. Kim, A. Prakash, D. Rus, J. Zhao, B. K. H. Low, and P. P. Liang, “Mem1: Learning to synergize memory and reasoning for efficient long-horizon agents,” *arXiv preprint arXiv:2506.15841*, 2025.
- [75] D. Edge, H. Trinh, N. Cheng, J. Bradley, A. Chao, A. Mody, S. Pruitt, D. Metropolitan, R. O. Ness, and J. Larson, “From local to global: A graph rag approach to query-focused summarization,” *arXiv preprint arXiv:2404.16130*, 2024.
- [76] B. Kynoch, H. Latapie, and D. van der Sluis, “Recallm: An adaptable memory mechanism with temporal understanding for large language models,” *arXiv preprint arXiv:2307.02738*, 2023.
- [77] X. Tang, T. Qin, T. Peng, Z. Zhou, D. Shao, T. Du, X. Wei, P. Xia, F. Wu, H. Zhu *et al.*, “Agent kb: Leveraging cross-domain experience for agentic problem solving,” *arXiv preprint arXiv:2507.06229*, 2025.
- [78] Z. Cai, X. Guo, Y. Pei, J. Feng, J. Su, J. Chen, Y.-Q. Zhang, W.-Y. Ma, M. Wang, and H. Zhou, “Flex: Continuous agent evolution via forward learning from experience,” *arXiv preprint arXiv:2511.06449*, 2025.
- [79] J. Sun, C. Xu, L. Tang, S. Wang, C. Lin, Y. Gong, L. Ni, H.-Y. Shum, and J. Guo, “Think-on-graph: Deep and responsible reasoning of large language model on knowledge graph,” in *The Twelfth International Conference on Learning Representations*, 2024.
- [80] Y. Xiao, C. Zhou, Q. Zhang, B. Li, Q. Li, and X. Huang, “Reliable reasoning path: Distilling effective guidance for llm reasoning with knowledge graphs,” *IEEE Transactions on Knowledge and Data Engineering*, 2026.

- [81] W. Zhong, L. Guo, Q. Gao, H. Ye, and Y. Wang, “Memorybank: Enhancing large language models with long-term memory,” in *Proceedings of the AAAI Conference on Artificial Intelligence*, 2024.
- [82] L. LUO, Y.-F. Li, G. Haffari, and S. Pan, “Reasoning on graphs: Faithful and interpretable large language model reasoning,” in *The Twelfth International Conference on Learning Representations*, 2024.
- [83] J. Kang, M. Ji, Z. Zhao, and T. Bai, “Memory os of ai agent,” *arXiv preprint arXiv:2506.06326*, 2025.
- [84] Z. Xu, R. Zhou, Y. Yin, H. Gao, M. Tomizuka, and J. Li, “Matrix: multi-agent trajectory generation with diverse contexts,” in *2024 IEEE International Conference on Robotics and Automation (ICRA)*, 2024.
- [85] Z. Gekhman, E. B. David, H. Orgad, E. Ofek, Y. Belinkov, I. Szpektor, J. Herzig, and R. Reichart, “Inside-out: Hidden factual knowledge in llms,” *arXiv preprint arXiv:2503.15299*, 2025.
- [86] G. Zhang, H. Ren, C. Zhan, Z. Zhou, J. Wang, H. Zhu, W. Zhou, and S. Yan, “Memevolve: Meta-evolution of agent memory systems,” *arXiv preprint arXiv:2512.18746*, 2025.
- [87] Moonshot AI, “Kimi k2.5,” 2026. [Online]. Available: <https://www.kimi.com/blog/kimi-k2-5.html>
- [88] A. Zhao, D. Huang, Q. Xu, M. Lin, Y.-J. Liu, and G. Huang, “Expel: Llm agents are experiential learners,” in *Proceedings of the AAAI Conference on Artificial Intelligence*, 2024.
- [89] C. Yang, Z. Sun, W. Wei, and W. Hu, “Beyond static summarization: Proactive memory extraction for llm agents,” *arXiv preprint arXiv:2601.04463*, 2026.
- [90] Y. Zhai, S. Tao, C. Chen, A. Zou, Z. Chen, Q. Fu, S. Mai, L. Yu, J. Deng, Z. Cao *et al.*, “Agentevolver: Towards efficient self-evolving agent system,” *arXiv preprint arXiv:2511.10395*, 2025.
- [91] B. J. Gutiérrez, Y. Shu, W. Qi, S. Zhou, and Y. Su, “From RAG to memory: Non-parametric continual learning for large language models,” in *Forty-second International Conference on Machine Learning*, 2025.
- [92] V. Markovic, L. Obradovic, L. Hajdu, and J. Pavlovic, “Optimizing the interface between knowledge graphs and llms for complex reasoning,” *arXiv preprint arXiv:2505.24478*, 2025.
- [93] J. Fang, X. Deng, H. Xu, Z. Jiang, Y. Tang, Z. Xu, S. Deng, Y. Yao, M. Wang, S. Qiao *et al.*, “Lightmem: Lightweight and efficient memory-augmented generation,” *arXiv preprint arXiv:2510.18866*, 2025.
- [94] A. Maharana, D.-H. Lee, S. Tulyakov, M. Bansal, F. Barbieri, and Y. Fang, “Evaluating very long-term conversational memory of LLM agents,” in *ACL*, 2024.
- [95] D. Wu, H. Wang, W. Yu, Y. Zhang, K.-W. Chang, and D. Yu, “LongMemEval: Benchmarking chat assistants on long-term interactive memory,” in *ICLR*, 2025.
- [96] Y. Hu, Y. Wang, and J. McAuley, “Evaluating memory in llm agents via incremental multi-turn interactions,” *arXiv preprint arXiv:2507.05257*, 2025.
- [97] D. Deshpande, V. Gangal, H. Mehta, A. Kannappan, R. Qian, and P. Wang, “Memtrack: Evaluating long-term memory and state tracking in multi-platform dynamic agent environments,” *arXiv preprint arXiv:2510.01353*, 2025.
- [98] J. He, L. Zhu, R. Wang, X. Wang, G. Haffari, and J. Zhang, “Madial-bench: Towards real-world evaluation of memory-augmented dialogue generation,” in *Proceedings of the 2025 Conference of the Nations of the Americas Chapter of the Association for Computational Linguistics: Human Language Technologies (Volume 1: Long Papers)*, 2025.
- [99] Z. Zhang, Q. Dai, L. Chen, Z. Jiang, R. Li, J. Zhu, X. Chen, Y. Xie, Z. Dong, and J.-R. Wen, “Memsim: A bayesian simulator for evaluating memory of llm-based personal assistants,” *arXiv preprint arXiv:2409.20163*, 2024.
- [100] B. Wu, W. Wang, L. Lihaoran, Y. Deng, Y. Li, J. Yu, and B. Wang, “Interpersonal memory matters: A new task for proactive dialogue utilizing conversational history,” in *Proceedings of the 29th Conference on Computational Natural Language Learning*, 2025.
- [101] J. Xu, A. Szlam, and J. Weston, “Beyond goldfish memory: Long-term open-domain conversation,” in *Proceedings of the 60th annual meeting of the association for computational linguistics (volume 1: long papers)*, 2022.
- [102] H. Xue, F. Tang, M. Hu, Y. Liu, Q. Huang, Y. Li, C. Liu, Z. Xu, C. Zhang, C.-M. Feng *et al.*, “Mmrc: A large-scale benchmark for understanding multimodal large language model in real-world conversation,” *arXiv preprint arXiv:2502.11903*, 2025.
- [103] H. Tan, Z. Zhang, C. Ma, X. Chen, Q. Dai, and Z. Dong, “Membench: Towards more comprehensive evaluation on the memory of llm-based agents,” *arXiv preprint arXiv:2506.21605*, 2025.
- [104] L. Wan and W. Ma, “Storybench: A dynamic benchmark for evaluating long-term memory with multi turns,” *arXiv preprint arXiv:2506.13356*, 2025.
- [105] J. Kim, W. Chay, H. Hwang, D. Kyung, H. Chung, E. Cho, Y. Jo, and E. Choi, “Dialsim: A real-time simulator for evaluating long-term multi-party dialogue understanding of conversation systems,” *arXiv preprint arXiv:2406.13144*, 2024.
- [106] H. Bian, Z. Yao, S. Hu, Z. Xu, S. Zhang, Y. Guo, Z. Yang, X. Han, H. Wang, and R. Chen, “Realmem: Benchmarking llms in real-world memory-driven interaction,” *arXiv preprint arXiv:2601.06966*, 2026.
- [107] B. Jiang, Z. Hao, Y.-M. Cho, B. Li, Y. Yuan, S. Chen, L. Ungar, C. J. Taylor, and D. Roth, “Know me, respond to me: Benchmarking llms for dynamic user profiling and personalized responses at scale,” *arXiv preprint arXiv:2504.14225*, 2025.
- [108] Y. Du, H. Wang, Z. Zhao, B. Liang, B. Wang, W. Zhong, Z. Wang, and K.-F. Wong, “Perltaq: A personal long-term memory dataset for memory classification, retrieval, and synthesis in question answering,” *arXiv preprint arXiv:2402.16288*, 2024.
- [109] Z. Zhang, Y. Zhang, H. Tan, R. Li, and X. Chen, “Explicit vs implicit memory: Exploring multi-hop complex reasoning over personalized information,” *arXiv preprint arXiv:2508.13250*, 2025.
- [110] S. Zhao, M. Hong, Y. Liu, D. Hazarika, and K. Lin, “Do LLMs recognize your preferences? evaluating personalized preference following in LLMs,” in *The Thirteenth International Conference on Learning Representations*, 2025.
- [111] Z. Jia, Q. Liu, H. Li, Y. Chen, and J. Liu, “Evaluating the long-term memory of large language models,” in *Findings of the Association for Computational Linguistics: ACL 2025*, 2025.
- [112] A. Miyai, Z. Zhao, K. Egashira, A. Sato, T. Sunada, S. Onohara, H. Yamamishi, M. Toyooka, K. Nishina, R. Maeda *et al.*, “Webchorearena: Evaluating web browsing agents on realistic tedious web tasks,” *arXiv preprint arXiv:2506.01952*, 2025.
- [113] Y. Deng, X. Zhang, W. Zhang, Y. Yuan, S. K. Ng, and T.-S. Chua, “On the multi-turn instruction following for conversational web agents,” in *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, 2024.
- [114] S. Yao, H. Chen, J. Yang, and K. Narasimhan, “Webshop: Towards scalable real-world web interaction with grounded language agents,” *Advances in Neural Information Processing Systems*, 2022.
- [115] S. Zhou, F. F. Xu, H. Zhu, X. Zhou, R. Lo, A. Sridhar, X. Cheng, T. Ou, Y. Bisk, D. Fried *et al.*, “Webarena: A realistic web environment for building autonomous agents,” in *12th International Conference on Learning Representations, ICLR 2024*, 2024.
- [116] S. Tian, Z. Zhang, L.-Y. Chen, and Z. Liu, “Mmina: Benchmarking multihop multimodal internet agents,” in *Findings of the Association for Computational Linguistics*, 2025.
- [117] T. Kwiatkowski, J. Palomaki, O. Redfield, M. Collins, A. Parikh, C. Alberti, D. Epstein, I. Polosukhin, J. Devlin, K. Lee *et al.*, “Natural questions: a benchmark for question answering research,” *Transactions of the Association for Computational Linguistics*, 2019.
- [118] M. Joshi, E. Choi, D. S. Weld, and L. Zettlemoyer, “Triviaqa: A large scale distantly supervised challenge dataset for reading comprehension,” in *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, 2017.
- [119] A. Mallen, A. Asai, V. Zhong, R. Das, D. Khashabi, and H. Hajishirzi, “When not to trust language models: Investigating effectiveness of parametric and non-parametric memories,” in *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, 2023.
- [120] Z. Yang, P. Qi, S. Zhang, Y. Bengio, W. Cohen, R. Salakhutdinov, and C. D. Manning, “Hotpotqa: A dataset for diverse, explainable multi-hop question answering,” in *Proceedings of the 2018 conference on empirical methods in natural language processing*, 2018.
- [121] X. Ho, A.-K. D. Nguyen, S. Sugawara, and A. Aizawa, “Constructing a multi-hop qa dataset for comprehensive evaluation of reasoning steps,” in *Proceedings of the 28th International Conference on Computational Linguistics*, 2020.
- [122] H. Trivedi, N. Balasubramanian, T. Khot, and A. Sabharwal, “Musique: Multihop questions via single-hop question composition,” *Transactions of the Association for Computational Linguistics*, 2022.
- [123] Y. Bai, X. Lv, J. Zhang, H. Lyu, J. Tang, Z. Huang, Z. Du, X. Liu, A. Zeng, L. Hou *et al.*, “Longbench: A bilingual, multitask benchmark for long context understanding,” in *Proceedings of the 62nd annual meeting of the association for computational linguistics (volume 1: Long papers)*, 2024.
- [124] Y. Bai, S. Tu, J. Zhang, H. Peng, X. Wang, X. Lv, S. Cao, J. Xu, L. Hou, Y. Dong *et al.*, “Longbench v2: Towards deeper understanding and reasoning on realistic long-context multitasks,” in *Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, 2025.

- [125] C.-P. Hsieh, S. Sun, S. Kriman, S. Acharya, D. Rekesh, F. Jia, and B. Ginsburg, “RULER: What’s the real context size of your long-context language models?” in *First Conference on Language Modeling*, 2024.
- [126] Y. Kuratov, A. Bulatov, P. Anokhin, I. Rodkin, D. Sorokin, A. Sorokin, and M. Burtsev, “Babilong: Testing the limits of llms with long context reasoning-in-a-haystack,” *Advances in Neural Information Processing Systems*, 2024.
- [127] H. Wang, H. Shi, S. Tan, W. Qin, W. Wang, T. Zhang, A. Nambi, T. Ganu, and H. Wang, “Multimodal needle in a haystack: Benchmarking long-context capability of multimodal large language models,” in *Proceedings of the 2025 Conference of the Nations of the Americas Chapter of the Association for Computational Linguistics: Human Language Technologies (Volume 1: Long Papers)*, 2025.
- [128] D. Chen, S. Niu, K. Li, P. Liu, X. Zheng, B. Tang, X. Li, F. Xiong, and Z. Li, “Halumem: Evaluating hallucinations in memory systems of agents,” *arXiv preprint arXiv:2511.03506*, 2025.
- [129] Q. Ai, Y. Tang, C. Wang, J. Long, W. Su, and Y. Liu, “Memorybench: A benchmark for memory and continual learning in llm systems,” *arXiv preprint arXiv:2510.17281*, 2025.
- [130] J. Zheng, X. Cai, Q. Li, D. Zhang, Z. Li, Y. Zhang, L. Song, and Q. Ma, “Lifelongagentbench: Evaluating llm agents as lifelong learners,” *arXiv preprint arXiv:2505.11942*, 2025.
- [131] C.-K. Wu, Z. R. Tam, C.-Y. Lin, Y.-N. Chen, and H.-y. Lee, “Streambench: Towards benchmarking continuous improvement of language agents,” *Advances in Neural Information Processing Systems*, 2024.
- [132] T. Wei, N. Sachdeva, B. Coleman, Z. He, Y. Bei, X. Ning, M. Ai, Y. Li, J. He, E. H. Chi *et al.*, “Evo-memory: Benchmarking llm agent test-time learning with self-evolving memory,” *arXiv preprint arXiv:2511.20857*, 2025.
- [133] K. Grauman, A. Westbury, E. Byrne, Z. Chavis, A. Furnari, R. Girdhar, J. Hamburger, H. Jiang, M. Liu, X. Liu *et al.*, “Ego4d: Around the world in 3,000 hours of egocentric video,” in *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, 2022.
- [134] J. Yang, S. Liu, H. Guo, Y. Dong, X. Zhang, S. Zhang, P. Wang, Z. Zhou, B. Xie, Z. Wang *et al.*, “Egolife: Towards egocentric life assistant,” in *Proceedings of the Computer Vision and Pattern Recognition Conference*, 2025.
- [135] M. Shridhar, X. Yuan, M.-A. Cote, Y. Bisk, A. Trischler, and M. Hausknecht, “{ALFW}orld: Aligning text and embodied environments for interactive learning,” in *International Conference on Learning Representations*, 2021.
- [136] M. Chevalier-Boisvert, D. Bahdanau, S. Lahouli, L. Willems, C. Saharia, T. H. Nguyen, and Y. Bengio, “BabyAI: First steps towards grounded language learning with a human in the loop,” in *International Conference on Learning Representations*, 2019.
- [137] R. Wang, P. Jansen, M.-A. Côté, and P. Ammanabrolu, “Scienceworld: Is your agent smarter than a 5th grader?” in *Proceedings of the 2022 Conference on Empirical Methods in Natural Language Processing*, 2022.
- [138] Z. Xi, Y. Ding, W. Chen, B. Hong, H. Guo, J. Wang, X. Guo, D. Yang, C. Liao, W. He *et al.*, “Agentgym: Evaluating and training large language model-based agents across diverse environments,” in *Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, 2025.
- [139] M. Chang, J. Zhang, Z. Zhu, C. Yang, Y. Yang, Y. Jin, Z. Lan, L. Kong, and J. He, “Agentboard: An analytical evaluation board of multi-turn llm agents,” *Advances in neural information processing systems*, 2024.
- [140] C. E. Jimenez, J. Yang, A. Wettig, S. Yao, K. Pei, O. Press, and K. R. Narasimhan, “SWE-bench: Can language models resolve real-world github issues?” in *The Twelfth International Conference on Learning Representations*, 2024.
- [141] G. Mialon, C. Fourrier, T. Wolf, Y. LeCun, and T. Scialom, “Gaia: a benchmark for general ai assistants,” in *The Twelfth International Conference on Learning Representations*, 2023.
- [142] K. Chen, Y. Ren, Y. Liu, X. Hu, H. Tian, T. Xie, F. Liu, H. Zhang, H. Liu, Y. Gong *et al.*, “xbench: Tracking agents productivity scaling with profession-aligned real-world evaluations,” *arXiv preprint arXiv:2506.13651*, 2025.
- [143] Y. Qin, S. Liang, Y. Ye, K. Zhu, L. Yan, Y. Lu, Y. Lin, X. Cong, X. Tang, B. Qian, S. Zhao, L. Hong, R. Tian, R. Xie, J. Zhou, M. Gerstein, dahai li, Z. Liu, and M. Sun, “ToolLLM: Facilitating large language models to master 16000+ real-world APIs,” in *The Twelfth International Conference on Learning Representations*, 2024.
- [144] B. Li, Z. Lin, D. Pathak, J. Li, Y. Fei, K. Wu, T. Ling, X. Xia, P. Zhang, G. Neubig *et al.*, “Genai-bench: Evaluating and improving compositional text-to-visual generation,” *arXiv preprint arXiv:2406.13743*, 2024.
- [145] Z. Huang, S. Gutierrez, H. Kamana, and S. MacNeil, “Memory sandbox: Transparent and interactive memory management for conversational agents,” in *Adjunct Proceedings of the 36th Annual ACM Symposium on User Interface Software and Technology*, 2023.
- [146] H. Li, C. Yang, A. Zhang, Y. Deng, X. Wang, and T.-S. Chua, “Hello again! Ilm-powered personalized agent for long-term dialogue,” in *Proceedings of the 2025 Conference of the Nations of the Americas Chapter of the Association for Computational Linguistics: Human Language Technologies (Volume 1: Long Papers)*, 2025.
- [147] Z. Pan, Q. Wu, H. Jiang, X. Luo, H. Cheng, D. Li, Y. Yang, C.-Y. Lin, H. V. Zhao, L. Qiu, and J. Gao, “Secom: On memory construction and retrieval for personalized conversational agents,” in *ICLR*, 2025.
- [148] Y. Dong, X. Jiang, J. Qian, T. Wang, K. Zhang, Z. Jin, and G. Li, “A survey on code generation with ilm-based agents,” *arXiv preprint arXiv:2508.00083*, 2025.
- [149] M. A. Islam, M. E. Ali, and M. R. Parvez, “Codesim: Multi-agent code generation and problem solving through simulation-driven planning and debugging,” *arXiv preprint arXiv:2502.05664*, 2025.
- [150] S. Hong, M. Zhuge, J. Chen, X. Zheng, Y. Cheng, J. Wang, C. Zhang, Z. Wang, S. K. S. Yau, Z. Lin *et al.*, “Metagpt: Meta programming for a multi-agent collaborative framework,” in *The Twelfth International Conference on Learning Representations*, 2023.
- [151] C. Qian, W. Liu, H. Liu, N. Chen, Y. Dang, J. Li, C. Yang, W. Chen, Y. Su, X. Cong *et al.*, “Chatdev: Communicative agents for software development,” in *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, 2024.
- [152] M.-T. Shen and Y.-J. Joung, “Talm: Dynamic tree-structured multi-agent framework with long-term memory for scalable code generation,” *arXiv preprint arXiv:2510.23010*, 2025.
- [153] N. Shinn, F. Cassano, A. Gopinath, K. Narasimhan, and S. Yao, “Reflexion: Language agents with verbal reinforcement learning,” *Advances in Neural Information Processing Systems*, 2023.
- [154] J. Guo, C. Wang, X. Xu, Z. Su, and X. Zhang, “Repoaudit: An autonomous llm-agent for repository-level code auditing,” *arXiv preprint arXiv:2501.18160*, 2025.
- [155] Q. Wang, Z. Cheng, S. Zhang, F. Liu, R. Xu, H. Lian, K. Wang, X. Yu, J. Yin, S. Hu *et al.*, “Memgovern: Enhancing code agents through learning from governed human experiences,” *arXiv preprint arXiv:2601.06789*, 2026.
- [156] A. Krishnamoorthy, K. Ivatury, and B. Ahmadnia, “Multi-agent reinforcement learning for interactive code debugging with human feedback and memory,” in *Proceedings of the 15th International Conference on Recent Advances in Natural Language Processing-Natural Language Processing in the Generative AI Era*, 2025.
- [157] S. Cai, J. Zhang, K. Bao, C. Gao, Q. Wang, F. Feng, and X. He, “Agentic feedback loop modeling improves recommendation and user simulation,” in *Proceedings of the 48th International ACM SIGIR conference on Research and Development in Information Retrieval*, 2025.
- [158] F. Liu, X. Lin, H. Yu, M. Wu, J. Wang, Q. Zhang, Z. Zhao, Y. Xia, Y. Zhang, W. Li *et al.*, “Recoworld: Building simulated environments for agentic recommender systems,” *arXiv preprint arXiv:2509.10397*, 2025.
- [159] J. Chen, “Memory assisted llm for personalized recommendation system,” *arXiv preprint arXiv:2505.03824*, 2025.
- [160] J. Liu, S. Gu, D. Li, G. Zhang, M. Han, H. Gu, P. Zhang, T. Lu, L. Shang, and N. Gu, “Agentcf++: Memory-enhanced llm-based agents for popularity-aware cross-domain recommendations,” in *Proceedings of the 48th International ACM SIGIR Conference on Research and Development in Information Retrieval*, 2025.
- [161] Y. Xi, W. Liu, J. Lin, B. Chen, R. Tang, W. Zhang, and Y. Yu, “Memocrs: Memory-enhanced sequential conversational recommender systems with large language models,” in *Proceedings of the 33rd ACM International Conference on Information and Knowledge Management*, 2024.
- [162] A. Zhang, Y. Chen, L. Sheng, X. Wang, and T.-S. Chua, “On generative agents in recommendation,” in *Proceedings of the 47th international ACM SIGIR conference on research and development in Information Retrieval*, 2024.
- [163] Y. Zhu, H. Steck, D. Liang, Y. He, N. Kallus, and J. Li, “Llm-based conversational recommendation agents with collaborative verbalized experience,” *Proceedings of the Proc. of EMNLP Findings*, 2025.
- [164] J. Zhang, Y. Hou, R. Xie, W. Sun, J. McAuley, W. X. Zhao, L. Lin, and J.-R. Wen, “Agentcf: Collaborative learning with autonomous language agents for recommender systems,” in *Proceedings of the ACM Web Conference 2024*, 2024.
- [165] Y. Wang, Z. Jiang, Z. Chen, F. Yang, Y. Zhou, E. Cho, X. Fan, Y. Lu, X. Huang, and Y. Yang, “Recmind: Large language model powered agent

- for recommendation,” in *Findings of the Association for Computational Linguistics: NAACL 2024*, 2024.
- [166] T. Guo, C. Liu, H. Wang, V. Mannam, F. Wang, X. Chen, X. Zhang, and C. K. Reddy, “Knowledge graph enhanced language agents for recommendation,” *arXiv preprint arXiv:2410.19627*, 2024.
- [167] J. Huang, X. Zou, L. Xia, and Q. Li, “Mr. rec: Synergizing memory and reasoning for personalized recommendation assistant with llms,” *arXiv preprint arXiv:2510.14629*, 2025.
- [168] H. Li, Y. Cao, Y. Yu, S. R. Javaji, Z. Deng, Y. He, Y. Jiang, Z. Zhu, K. Subbalakshmi, J. Huang *et al.*, “Investorbench: A benchmark for financial decision-making tasks with llm-based agent,” in *Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, 2025.
- [169] Y. Li, Y. Yu, H. Li, Z. Chen, and K. Khashanah, “Tradinggpt: Multi-agent system with layered memory and distinct characters for enhanced financial trading performance,” *arXiv preprint arXiv:2309.03736*, 2023.
- [170] Y. Yu, Z. Yao, H. Li, Z. Deng, Y. Jiang, Y. Cao, Z. Chen, J. Suchow, Z. Cui, R. Liu *et al.*, “Fincon: A synthesized llm multi-agent system with conceptual verbal reinforcement for enhanced financial decision making,” *Advances in Neural Information Processing Systems*, 2024.
- [171] W. Zhang, L. Zhao, H. Xia, S. Sun, J. Sun, M. Qin, X. Li, Y. Zhao, Y. Zhao, X. Cai *et al.*, “A multimodal foundation agent for financial trading: Tool-augmented, diversified, and generalist,” in *Proceedings of the 30th ACM SIGKDD conference on knowledge discovery and data mining*, 2024.
- [172] Z. Wang, S. Cai, A. Liu, Y. Jin, J. Hou, B. Zhang, H. Lin, Z. He, Z. Zheng, Y. Yang *et al.*, “Jarvis-1: Open-world multi-task agents with memory-augmented multimodal language models,” *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2024.
- [173] M. A. Raad, A. Ahuja, C. Barros, F. Besse, A. Bolt, A. Bolton, B. Brownfield, G. Buttimore, M. Cant, S. Chakera *et al.*, “Scaling instructable agents across many simulated worlds,” *arXiv preprint arXiv:2404.10179*, 2024.
- [174] A. Bolton, A. Lerchner, A. Cordell, A. Moufarek, A. Bolt, A. Lampinen, A. Mitenkova, A. O. Hallingstad, B. Vujatovic, B. Li *et al.*, “Sima 2: A generalist embodied agent for virtual worlds,” *arXiv preprint arXiv:2512.04797*, 2025.
- [175] G. Sarch, Y. Wu, M. Tarr, and K. Fragkiadaki, “Open-ended instructable embodied agents with memory-augmented large language models,” in *Findings of the Association for Computational Linguistics: EMNLP 2023*, 2023.
- [176] R. Li, W. Guo, Z. Wu, C. Wang, H. Deng, Z. Weng, Y.-P. Tan, and Z. Wang, “Map-VLA: Memory-augmented prompting for vision-language-action model in robotic manipulation,” *arXiv preprint arXiv:2511.09516*, 2025.
- [177] M. Memmel, J. Berg, B. Chen, A. Gupta, and J. Francis, “STRAP: Robot sub-trajectory retrieval for augmented policy learning,” in *ICLR*, 2025.
- [178] J. Liu, Y. Qi, J. Zhang, M. Li, S. Wang, K. Wu, H. Ye, H. Zhang, Z. Chen, F. Zhong *et al.*, “TrackVLA++: Unleashing reasoning and memory capabilities in VLA models for embodied visual tracking,” *arXiv preprint arXiv:2510.07134*, 2025.
- [179] S. Schmidgall, R. Ziaeい, C. Harris, E. Reis, J. Jopling, and M. Moor, “Agentclinic: A multimodal agent benchmark to evaluate AI in simulated clinical environments,” *arXiv preprint arXiv:2405.07960*, 2024.
- [180] J. Hu, A. Wang, Q. Xie, H. Ma, Z. Li, and D. Guo, “Agentmental: An interactive multi-agent framework for explainable and adaptive mental health assessment,” *arXiv preprint arXiv:2508.11567*, 2025.
- [181] J. Li, Y. Lai, W. Li, J. Ren, M. Zhang, X. Kang, S. Wang, P. Li, Y.-Q. Zhang, W. Ma *et al.*, “Agent hospital: A simulacrum of hospital with evolvable medical agents,” *arXiv preprint arXiv:2405.02957*, 2024.
- [182] O. Bodenreider, “The unified medical language system (umls): integrating biomedical terminology,” *Nucleic acids research*, 2004.
- [183] M. Moritz, E. Topol, and P. Rajpurkar, “Coordinated ai agents for advancing healthcare,” *Nature Biomedical Engineering*, 2025.
- [184] B. P. de Almeida, G. Richard, H. Dalla-Torre, C. Blum, L. Hexemer, P. Pandey, S. Laurent, C. Rajesh, M. Lopez, A. Laterre *et al.*, “A multimodal conversational agent for dna, rna and protein tasks,” *Nature Machine Intelligence*, 2025.
- [185] A. M. Bran, S. Cox, O. Schilter, C. Baldassari, A. D. White, and P. Schwaller, “Augmenting large language models with chemistry tools,” *Nature Machine Intelligence*, 2024.
- [186] Y. Zou, A. H. Cheng, A. Aldossary, J. Bai, S. X. Leong, J. A. Campos-Gonzalez-Angulo, C. Choi, C. T. Ser, G. Tom, A. Wang *et al.*, “El agente: An autonomous agent for quantum chemistry,” *Matter*, 2025.
- [187] K. Huang, S. Zhang, H. Wang, Y. Qu, Y. Lu, Y. Roohani, R. Li, L. Qiu, G. Li, J. Zhang *et al.*, “Biomni: A general-purpose biomedical ai agent,” *biorxiv*, 2025.
- [188] Z. Zhang, Z. Ren, C.-W. Hsu, W. Chen, Z.-W. Hong, C.-F. Lee, A. Penn, H. Xu, D. J. Zheng, S. Miao *et al.*, “A multimodal robotic platform for multi-element electrocatalyst discovery,” *Nature*, 2025.
- [189] K. Swanson, W. Wu, N. L. Bulaong, J. E. Pak, and J. Zou, “The virtual lab of AI agents designs new SARS-CoV-2nanobodies,” *Nature*, 2025.
- [190] Z. Xiong, Y. Lin, W. Xie, P. He, J. Tang, H. Lakkaraju, and Z. Xiang, “How memory management impacts llm agents: An empirical study of experience-following behavior,” *arXiv preprint arXiv:2505.16067*, 2025.
- [191] Y. Bei, W. Zhang, S. Wang, W. Chen, S. Zhou, H. Chen, Y. Li, J. Bu, S. Pan, Y. Yu *et al.*, “Graphs meet ai agents: Taxonomy, progress, and future opportunities,” *arXiv preprint arXiv:2506.18019*, 2025.
- [192] Y. Du, W. Huang, D. Zheng, Z. Wang, S. Montella, M. Lapata, K.-F. Wong, and J. Z. Pan, “Rethinking memory in ai: Taxonomy, operations, topics, and future directions,” *arXiv preprint arXiv:2505.00675*, 2025.
- [193] J. Leskovec, K. J. Lang, A. Dasgupta, and M. W. Mahoney, “Community structure in large networks: Natural cluster sizes and the absence of large well-defined clusters,” *Internet Mathematics*, 2009.
- [194] W. Fan, C. Hu, and C. Tian, “Incremental graph computations: Doable and undoable,” in *Proceedings of the 2017 ACM International Conference on Management of Data*, 2017.
- [195] Y. A. Malkov and D. A. Yashunin, “Efficient and robust approximate nearest neighbor search using hierarchical navigable small world graphs,” *IEEE transactions on pattern analysis and machine intelligence*, 2018.
- [196] J. Ahn, S. Hong, S. Yoo, O. Mutlu, and K. Choi, “A scalable processing-in-memory accelerator for parallel graph processing,” in *Proceedings of the 42nd annual international symposium on computer architecture*, 2015.
- [197] Y. Shao, H. Li, X. Gu, H. Yin, Y. Li, X. Miao, W. Zhang, B. Cui, and L. Chen, “Distributed graph neural network training: A survey,” *ACM Computing Surveys*, 2024.
- [198] T. T. Mueller, D. Usynin, J. C. Paetzold, D. Rueckert, and G. Kaisis, “Sok: Differential privacy on graph-structured data,” *arXiv preprint arXiv:2203.09205*, 2022.
- [199] A. Zou, Z. Wang, N. Carlini, M. Nasr, J. Z. Kolter, and M. Fredrikson, “Universal and transferable adversarial attacks on aligned language models,” *arXiv preprint arXiv:2307.15043*, 2023.
- [200] N. Carlini, M. Jagielski, C. A. Choquette-Choo, D. Paleka, W. Pearce, H. Anderson, A. Terzis, K. Thomas, and F. Tramèr, “Poisoning web-scale training datasets is practical,” in *2024 IEEE Symposium on Security and Privacy (SP)*, 2024.
- [201] C. Finn, P. Abbeel, and S. Levine, “Model-agnostic meta-learning for fast adaptation of deep networks,” in *ICML*, 2017.
- [202] A. Hogan, E. Blomqvist, M. Cochez, C. d’Amato, G. D. Melo, C. Gutierrez, S. Kirrane, J. E. L. Gayo, R. Navigli, S. Neumaier *et al.*, “Knowledge graphs,” *ACM Computing Surveys*, 2021.
- [203] P. W. Battaglia, J. B. Hamrick, V. Bapst, A. Sanchez-Gonzalez, V. Zambaldi, M. Malinowski, A. Tacchetti, D. Raposo, A. Santoro, R. Faulkner *et al.*, “Relational inductive biases, deep learning, and graph networks,” *arXiv preprint arXiv:1806.01261*, 2018.
- [204] J. Kaplan, S. McCandlish, T. Henighan, T. B. Brown, B. Chess, R. Child, S. Gray, A. Radford, J. Wu, and D. Amodei, “Scaling laws for neural language models,” *arXiv preprint arXiv:2001.08361*, 2020.
- [205] D. Hassabis, D. Kumaran, C. Summerfield, and M. Botvinick, “Neuroscience-inspired artificial intelligence,” *Neuron*, 2017.

APPENDIX A PRELIMINARIES

This section provides foundational concepts and formal definitions necessary for understanding graph-based agent memory systems. Since graphs serve as the fundamental structure, we begin with graph theory fundamentals, then introduce LLM-based agent architectures, and finally formalize memory components. Before we give the formal definitions, the frequent symbols are listed in Table II

A. Graph Foundations

a) *Graph Definition:* We define a graph as $G = (V, E, X)$, where V denotes a set of nodes, $E \subseteq V \times V$ denotes a set of edges encoding pairwise relations between nodes, and X denotes node-associated features.

The graph structure is represented by an adjacency matrix $A \in \mathbb{R}^{|V| \times |V|}$, where each entry A_{ij} characterizes the relation between nodes v_i and v_j . If $A_{ij} \in \{0, 1\}$, the graph is unweighted and indicates the absence or presence of an edge. If $A_{ij} \in \mathbb{R}$, the graph is weighted, and the corresponding edge weight is denoted by w_{ij} .

Node features X may consist of continuous vectors or unstructured texts. In text-attributed graphs, X corresponds to textual descriptions associated with nodes, and edge relations are typically binary and undirected, such that $e_{ij} = e_{ji}$. In knowledge graphs, node features may be represented as texts or vectors, while edges encode semantic relations and are generally directed, such that $e_{ij} \neq e_{ji}$. Knowledge graphs extensively use edge labels to represent relationship types (e.g., "located_in," "has_property"). When edge weights are present, they further quantify the strength or confidence of relations.

1) Different Graphs:

a) *Knowledge Graph:* A **knowledge graph** is an instantiation of the unified graph representation $G = (V, E, X)$ in which nodes correspond to entities and edges encode typed semantic relations. It is commonly formalized as $\mathcal{G} = (\mathcal{E}, \mathcal{R}, \mathcal{T})$, where \mathcal{E} denotes entities, \mathcal{R} denotes relation types, and $\mathcal{T} \subseteq \mathcal{E} \times \mathcal{R} \times \mathcal{E}$ is a set of relational triples. Each triple (h, r, t) corresponds to a directed edge e_{ij} from $v_i = h$ to $v_j = t$, with relation semantics attached to the edge. Knowledge graphs are typically directed and multi-relational, with node features X represented as textual descriptions, learned embeddings, or both.

b) *Temporal Graph:* A **temporal graph** extends the static graph $G = (V, E, X)$ by associating edges with time-dependent information drawn from a time domain T . Each edge $e_{ij} \in E$ is linked to a timestamp or interval $\tau(e_{ij}) \subseteq T$, enabling the adjacency structure A_{ij} to vary over time. This formulation supports dynamic neighborhood definitions and temporal reasoning over evolving interactions, event sequences, and state transitions, while preserving the same node feature representation X .

c) *Hypergraph:* A **hypergraph** generalizes the pairwise edge structure of $G = (V, E, X)$ by allowing each edge to connect more than two nodes. Formally, edges are defined as subsets of vertices, i.e., $E \subseteq 2^V$, enabling the representation of higher-order and multi-way relations that cannot be reduced to binary interactions. Node features X remain associated with

individual vertices, while neighborhood relations are induced by shared hyperedge membership rather than pairwise adjacency.

d) *Graph Variants:* Several commonly used graph variants can be viewed as task-driven instantiations of the unified representation $G = (V, E, X)$, obtained by imposing specific constraints on node and edge definitions. **Binary graphs** restrict edge values to $A_{ij} \in \{0, 1\}$ and model relation existence only. **Text-attributed graphs** associate nodes with textual features, enabling language-aware representation learning. **Chunk-based graphs** further decompose text into finer-grained units as nodes, with edges encoding contextual or semantic connections. **Hierarchical graphs** introduce structural constraints to represent multi-level relationships. These variants share the same underlying graph formulation and differ only in how V , E , and X are instantiated for practical modeling purposes. These graph variants provide flexible abstractions for retrieval, representation learning, and classification. In agent graph memory systems, they support efficient organization and reasoning over heterogeneous information while maintaining a unified structural foundation.

2) Graph Algorithms:

a) *Graph Embeddings:* Graph embeddings aim to encode nodes into continuous vector spaces while preserving graph structural or semantic information. Given a graph $G = (V, E, X)$, node embeddings are defined as a mapping

$$f : V \rightarrow \mathbb{R}^d,$$

where each node $v_i \in V$ is associated with a d -dimensional representation $\mathbf{h}_i = f(v_i)$.

b) *Topology-based embeddings:* A representative traditional approach is *Node2Vec*, which learns node embeddings by optimizing a Skip-gram objective over node sequences generated by biased random walks. Formally, the embedding \mathbf{h}_i is learned by maximizing $\log \Pr(v_j | v_i)$:

$$\Pr(v_j | v_i) = \frac{\exp(\mathbf{h}_i^\top \mathbf{h}_j)}{\sum_{v_k \in \mathcal{N}_l(v_i)} \exp(\mathbf{h}_i^\top \mathbf{h}_k)},$$

where $\mathcal{N}_l(v_i)$ denotes the context nodes of v_i induced by biased random walks of length l . This formulation captures graph structure through node co-occurrence patterns.

c) *GNN-based embeddings:* Graph Neural Networks compute node embeddings via iterative neighborhood aggregation. At layer k , the embedding of node v_i is updated as

$$\mathbf{h}_i^{(k+1)} = \sigma \left(\mathbf{W}^{(k)} \cdot \text{AGG}^{(k)} \left(\{\mathbf{h}_j^{(k)} | v_j \in \mathcal{N}(v_i)\} \right) \right),$$

where $\text{AGG}(\cdot)$ is a permutation-invariant aggregation function, $\mathbf{W}^{(k)}$ is a learnable parameter matrix, and $\sigma(\cdot)$ denotes a non-linear activation.

LM-based embeddings. For a text-attributed node v_i with associated token sequence X_i , a Transformer-based language model encodes the input as

$$\mathbf{H}_i = \text{Transformer}(X_i),$$

where $\mathbf{H}_i \in \mathbb{R}^{L \times d}$ denotes the final-layer token representations. The node embedding is defined as

$$\mathbf{h}_i = \mathbf{H}_{i,[\text{CLS}]},$$

i.e., the representation of the special classification token, following standard practice in BERT-style models.

TABLE II: Notations for Graph Foundations

Symbol	Formal Definition	Meaning
G	$G = (V, E, X)$	Graph represented by node set, edge set, and node features
V	$V = \{v_1, \dots, v_{ V }\}$	Set of nodes (vertices)
E	$E \subseteq V \times V$	Set of edges encoding pairwise relations between nodes
X	$X \in \mathcal{X}$	Node feature set, consisting of vectors or unstructured texts
v_i, v_j	$v_i, v_j \in V$	The i -th and j -th nodes
e_{ij}	$e_{ij} = (v_i, v_j)$	Edge from node v_i to node v_j
$\mathcal{N}(v_i)$	$\mathcal{N}(v_i) = \{v_j \mid e_{ij} \in E\}$	Neighborhood set of node v_i
$d(v_i)$	$d(v_i) = \mathcal{N}(v_i) $	Degree of node v_i
A	$A \in \mathbb{R}^{ V \times V }$	Adjacency matrix representing graph structure
A_{ij}	$A_{ij} \in \{0, 1\}$ or \mathbb{R}	Adjacency matrix entry encoding the relation between v_i and v_j
w_{ij}	$w_{ij} \in \mathbb{R}$	Weight associated with edge e_{ij} (if applicable)
\mathbf{h}_i	$\mathbf{h}_i \in \mathbb{R}^d$	d -dimensional embedding of node v_i

d) Graph Traversal: Graph traversal methods define systematic or stochastic procedures for exploring the topology of a graph $G = (V, E)$, with the goal of discovering reachable nodes, structural paths, or task-relevant subgraphs that support retrieval, reasoning, and representation learning.

- **Breadth-first search (BFS)** explores the graph in increasing order of shortest-path distance from a source node v_i , iteratively visiting nodes in successive neighborhoods $N_1(v_i), N_2(v_i), \dots$, and is commonly used for local neighborhood expansion and shortest-path discovery in unweighted graphs.
- **Depth-first search (DFS)** recursively explores a path by traversing adjacent nodes as deeply as possible before backtracking, enabling efficient enumeration of paths, connectivity analysis, and cycle detection.
- **Random walk** defines a stochastic traversal process where a node sequence (v_0, v_1, \dots, v_k) is generated according to transition probabilities

$$p(v_{j+1} = v \mid v_j) = \frac{A_{v_j v}}{\sum_{u \in N(v_j)} A_{v_j u}},$$

and serves as the theoretical foundation for classical node embedding methods.

- **Shortest-path traversal** seeks a path $P = \langle v_0 = s, \dots, v_k = t \rangle$ that minimizes the accumulated edge weight

$$\sum_{i=0}^{k-1} w_{v_i v_{i+1}},$$

where w_{ij} denotes the weight associated with edge e_{ij} .

- **Subgraph extraction** identifies a local induced subgraph centered at v_i , typically defined as the k -hop neighborhood

$$N_k(v_i) = \{v_j \in V \mid d(v_i, v_j) \leq k\},$$

which is widely used for localized retrieval and downstream graph-based modeling.

B. LLM-based Agents

Table III summarizes the key symbols used in the agent system and graph-based memory formulation, together with their semantic interpretations.

Symbol	Description
\mathcal{A}	LLM-based agent
\mathcal{S}	Environment state space
s_t	Environment state at time step t
Q	Task specification
t	Discrete interaction time step
a_t	Action selected by the agent at time t
o_t	Partial observation received by the agent at time t
r_t	Feedback or reward signal at time t
h_t	Agent interaction history up to time t
Ψ	Environment transition process
$O(\cdot)$	Observation function
\mathcal{M}_θ	Parameterized LLM policy with parameters θ
\mathcal{M}	Agent memory module
c_t	Retrieved memory context at time t
q_t	Memory query derived from task or observation
\mathcal{M}_G	Graph-based memory module
G_t	Memory graph at time t
V_t	Set of memory nodes at time t
E_t	Set of edges (relations) at time t
X_t	Node and edge attributes of the memory graph
v_i	A memory node
e_{ij}	A directed or undirected relation between v_i and v_j
\mathcal{D}	Static corpus/experience set for memory construction
Δ_t	Online memory update signal (o_t, a_t, r_t)

TABLE III: Notation used in the agent interaction loop and graph-based memory framework.

1) Agent System: An **LLM-based agent** is a decision-making system that employs a large language model as its central reasoning component to interact with an environment and accomplish a given task. Formally, an agent \mathcal{A} operates over an environment with state space \mathcal{S} and task specification Q .

At each time step t , the agent selects an action a_t that drives the evolution of the environment via a transition process Ψ , based on partial observations o_t rather than full access to the underlying state. The agent's behavior is governed by a policy induced by a parameterized language model, which integrates observations, interaction history h_t , and optionally external memory or tools to support reasoning and decision-making, while the concrete operational procedures are specified by the agent interaction loop introduced subsequently.

a) Agent Interaction Loop: Specifically, during the time step t , the agent interacts with the environment through a structured perception–retrieval–reasoning–action–update loop:

- **Perception.** The agent receives an observation

$$o_t = O(s_t, h_t, Q),$$

which encodes partial information about the environment state s_t under task specification Q .

- **Retrieval.** Given the current observation and interaction history, the agent queries its external memory:

$$c_t = \text{Retrieve}(\mathcal{M}, o_t, h_t),$$

where c_t denotes the retrieved contextual information.

- **Reasoning.** The LLM backbone integrates the observation, retrieved memory, and history to perform reasoning and decision-making:

$$a_t \sim \mathcal{M}_\theta(o_t, c_t, h_t).$$

- **Action.** The selected action a_t is executed in the environment, inducing a state transition

$$s_{t+1} \sim \Psi(s_{t+1} | s_t, a_t).$$

- **Update.** The agent incorporates new experience and feedback into memory:

$$\mathcal{M} \leftarrow \text{Update}(\mathcal{M}, o_t, a_t, r_t).$$

The memory module \mathcal{M} stores a collection of experience tuples and knowledge representations, and supports retrieval and update operations. Its internal structure is left unspecified at this stage and may be instantiated as a key-value store, episodic buffer, or structured graph memory in later sections.

2) *Prompt Construction from Memory:* In LLM-based agent systems, memory influences agent behavior primarily through prompt conditioning. At each time step t , the agent constructs a composite prompt that integrates system-level instructions, retrieved memory content, and the current task context. Formally, the prompt can be abstracted as

$$\text{Prompt}_t = \text{System}(I) \oplus \text{Memory}(c_t) \oplus \text{Task}(o_t),$$

where I specifies the agent’s role and operational constraints, c_t denotes the retrieved memory context, and o_t represents the current observation or query.

The memory context c_t is obtained by querying the memory module with a task-dependent query q_t , typically through similarity-based retrieval:

$$c_t = \text{TopK}_{m_i \in \mathcal{M}}(\text{sim}(q_t, m_i)),$$

where m_i denotes individual memory units and $\text{sim}(\cdot, \cdot)$ is a similarity function defined over their representations. This formulation provides a unified interface through which stored experiences and knowledge are incorporated into the agent’s reasoning process.

C. Graph-based Memory

In LLM-based agent systems, memory serves as a persistent mechanism for storing, organizing, and retrieving past experiences and knowledge. When memory is instantiated in a structured form, it can be naturally represented as a graph, which enables relational modeling, efficient retrieval, and incremental updates over time.

1) *Memory as Graph Format:* We formalize graph-based agent memory as a dynamic attributed graph

$$\mathcal{M}_G \triangleq G_t = (V_t, E_t, X_t),$$

where V_t denotes the set of memory nodes, $E_t \subseteq V_t \times V_t$ denotes the set of relations between nodes, and X_t represents node- and edge-associated attributes. This formulation is consistent with the basic graph foundation introduced earlier, while allowing task-specific instantiations.

In details, each node $v_i \in V_t$ corresponds to a memory unit, such as an entity, event, concept, or textual chunk. Each edge $e_{ij} \in E_t$ represents a relation between memory units, which may encode semantic, temporal, or causal dependencies. Node attributes $X_t(v_i)$ typically include textual content or vector embeddings, while edge attributes may include relation types, confidence scores, or temporal information.

2) Graph Memory Mechanism:

a) *Graph Memory Construction:* Memory construction refers to the initial formation of a graph-structured memory from unstructured or semi-structured information, independent of online agent actions. Formally, given a corpus or experience set \mathcal{D} , graph construction defines a mapping

$$\text{Construct} : \mathcal{D} \rightarrow G_0 = (V_0, E_0, X_0),$$

where nodes V_0 are extracted memory units and edges E_0 encode relations among them.

Construction typically involves (i) node extraction, where entities, events, concepts, or textual chunks are identified as nodes, and (ii) relation extraction, where semantic, temporal, or structural relations are identified to form edges. The resulting graph may be represented as triples (v_i, e_{ij}, v_j) and serves as the initial memory state.

b) *Graph Memory Retrieval:* Memory retrieval corresponds to querying the graph to identify a relevant subset of nodes or subgraphs:

$$\text{Retrieve} : (q, G_t) \rightarrow \mathcal{S}_t \subseteq V_t,$$

where the query q may be textual, structural, or embedding-based. Retrieval can be realized via semantic similarity over node representations, graph traversal from query-relevant nodes, or their combination.

c) *Graph Memory Update:* Memory update describes the online evolution of an existing memory graph driven by the agent’s interaction with the environment. At time step t , the update signal

$$\Delta_t \triangleq (o_t, a_t, r_t)$$

is derived from the agent’s observation, action, and feedback. Given the current memory graph G_t , the update operation is defined as

$$\text{Update} : (G_t, \Delta_t) \rightarrow G_{t+1}.$$

The update process integrates newly observed information into the graph by adding or modifying nodes and edges, adjusting their attributes, or revising relations to reflect newly acquired evidence. Unlike memory construction, which forms an initial graph from static data, memory update operates incrementally and enables continual adaptation of the memory structure during agent interaction.

TABLE IV: Comparison of Open-sourced Libraries for Graph-based Memory Systems

ID	Library	License	Apache2.0	MIT	External Construct.	Interaction Construct.	Graph Memory	Retrieval	Lifecycle	Temporality	Reasoning	Conditioning	Personalization	Hierarchy	Agent Integration
1	Cognee ³		✓		✓			✓							✓
2	LangMem ⁴			✓			✓	✓	✓		✓		✓		✓
3	Mem0 ⁵		✓			✓	✓	✓	✓		✓		✓	✓	✓
4	LightMem ⁶			✓			✓	✓	✓		✓		✓	✓	✓
5	O-Mem ⁷		✓			✓		✓	✓		✓		✓	✓	✓
6	OpenMemory ⁸		✓		✓	✓	✓	✓	✓	✓		✓		✓	✓
7	Memori ⁹		✓			✓		✓			✓				
8	MemMaching ¹⁰		✓				✓				✓				
9	Memary ¹¹			✓		✓	✓	✓	✓			✓	✓		✓
10	Graphiti ¹²		✓		✓	✓	✓	✓	✓		✓				
11	Memivid ¹³		✓		✓	✓		✓	✓	✓	✓				

3) *Graph Memory Quality*: To systematically evaluate graph-based memory in agent systems, it is necessary to assess both the quality of memory retrieval and the structural properties induced by the graph formulation, as well as their impact on downstream task performance. We summarize representative evaluation criteria along three complementary dimensions.

a) *Retrieval Effectiveness*: Retrieval quality measures the ability of the memory graph to surface relevant information in response to a query. Common metrics include Precision@K and Recall@K, which quantify relevance among the top-ranked retrieved nodes, and Mean Reciprocal Rank (MRR), which captures the ranking position of the first relevant memory.

b) *Graph Structural Quality*: Graph quality metrics evaluate whether the constructed memory graph provides a coherent and faithful representation of stored knowledge. Typical criteria include coherence, which reflects structural consistency and connectivity; completeness, which measures coverage of salient information; redundancy, which captures unnecessary duplication; and temporal consistency, which assesses whether temporal relations are correctly preserved.

c) *Task-level Utility*: Task performance metrics evaluate the functional usefulness of graph memory in agent decision-making, including task success rate, interaction efficiency, and generalization to unseen tasks or domains.

APPENDIX B OPEN-SOURCED LIBRARIES

In Table IV, we provide a systematic comparison of eleven representative open-source memory libraries across key functional dimensions. The columns capture essential aspects of memory systems, including license type, construction mode (external knowledge-based or interaction-driven), support for graph-based memory, retrieval mechanisms, lifecycle management, temporal modeling, reasoning capabilities, conditioning, personalization, hierarchical structure, and integration with agent frameworks. This structured overview allows for a nuanced comparison of memory design choices in agent memory design.

³<https://docs.cognee.ai/>

⁴<https://langchain-ai.github.io/langmem/>

⁵<https://docs.mem0.ai/open-source/overview>

⁶<https://github.com/zjunlp/LightMem>

⁷<https://github.com/OPPO-PersonalAI/O-Mem>

⁸<https://github.com/CaviraOSS/OpenMemory>

⁹<https://github.com/GibsonAI/Memori>

¹⁰<https://github.com/MemMachine/MemMachine>

¹¹<https://github.com/kingjulio8238/Memory>

¹²<https://github.com/getzep/graphiti>

¹³<https://github.com/memvid/memvid>